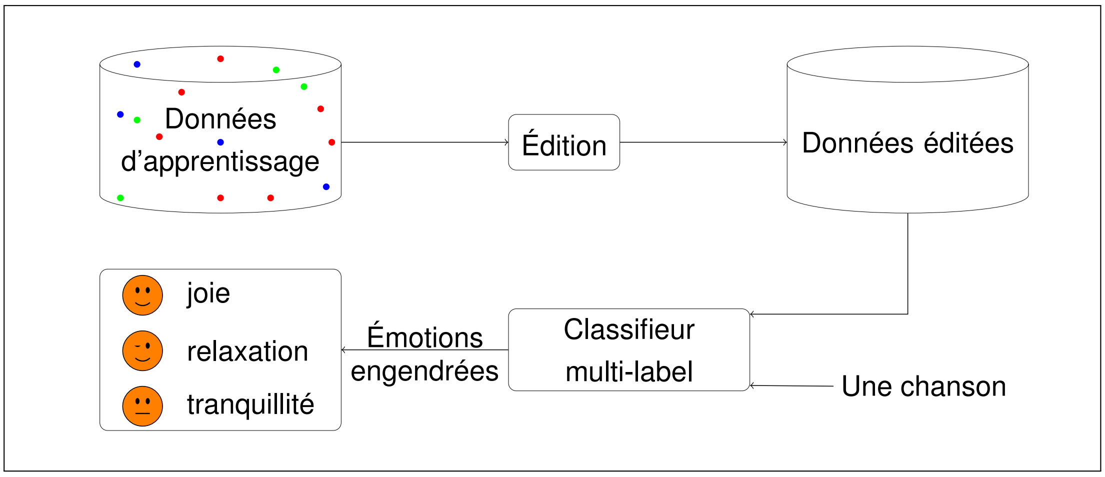
par Sawsan KANJ

Méthodes d’apprentissage pour la classification multi-label

Thèse présentée
pour l’obtention du grade
de Docteur de l’UTC

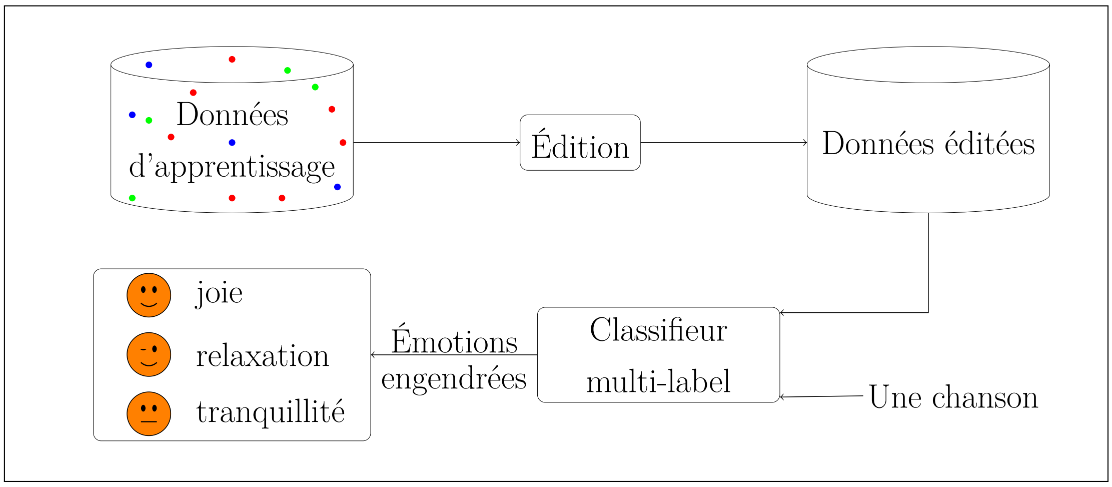


Soutenue le : 06 mai 2013
Spécialité : Technologie de l’Information et des Systèmes

Méthodes d'apprentissage pour la classification multi-label

Learning methods for multi-label classification

Sawsan KANJ



Thèse soutenue le 06 mai 2013 devant le jury composé de:

M. Fahed ABDALLAH	Maître de conférences, HDR (UTC)	Directeur de thèse
M. Olivier DEBEIR	Professeur (ULB)	Rapporteur
M. Thierry DENŒUX	Professeur des Universités (UTC)	Directeur de thèse
M. Lionel FILLATRE	Professeur (UNS)	Rapporteur
M. Mohamad KHALIL	Professeur (UL)	Examinateur
M. Walter SCHÖN	Professeur des Universités (UTC)	Examinateur
M. Kifah TOUT	Professeur (UL)	Directeur de thèse

To my lovely family

Résumé

La classification multi-label est une extension de la classification traditionnelle dans laquelle les classes ne sont pas mutuellement exclusives, chaque individu pouvant appartenir à plusieurs classes simultanément. Ce type de classification est requis par un grand nombre d'applications actuelles telles que la classification d'images et l'annotation de vidéos. Le principal objectif de cette thèse est la proposition de nouvelles méthodes pour répondre au problème de classification multi-label. La première partie de cette thèse s'intéresse au problème d'apprentissage multi-label dans le cadre des fonctions de croyance. Nous développons une méthode capable de tenir compte des corrélations entre les différentes classes et de classer les individus en utilisant le formalisme de représentation de l'incertitude pour les variables multi-valuées. La deuxième partie aborde le problème de l'édition des bases d'apprentissage pour la classification multi-label. Nous proposons un algorithme basé sur l'approche des k-plus proches voisins qui permet de détecter les exemples erronés dans l'ensemble d'apprentissage. Des expérimentations menées sur des jeux de données synthétiques et réelles montrent l'intérêt des approches étudiées.

Abstract

Multi-label classification is an extension of traditional single-label classification, where classes are not mutually exclusive, and each example can be assigned to several classes simultaneously. It is encountered in various modern applications such as scene classification and video annotation. The main objective of this thesis is the development of new techniques to address the problem of multi-label classification. The first part of this manuscript studies the problem of multi-label classification in the context of the theory of belief functions. We propose a multi-label learning method that is able to take into account relationships between labels and to classify new instances using the formalism of uncertainty representation for set-valued variables. The second part deals with the problem of prototype selection in the framework of multi-label learning. We propose an editing algorithm based on the k-nearest neighbors rule in order to purify training datasets and improve the performances of multi-label classification algorithms. Experimental results on synthetic and real-world datasets show the effectiveness of our approaches.

Acknowledgements

First and foremost, I would like to express my gratitude to my supervisors Mr. Fahed ABDALLAH, Mr. Thierry DENŒUX and Mr. Kifah TOUT, who have continuously supported my PhD study and research. Their enthusiasm, advice and encouragement guided the way for this long journey. I was fortunate to have them as my supervisors.

Some special thanks go to my two referees, Mr. Olivier DEBEIR, Professor at the Université Libre de Bruxelles and Mr. Lionel FILLATRE, Professor at the Université de Sophia-Antipolis, for their careful study of this dissertation and their valuable feedback.

I am further indebted to Mr. Mohamad KHALIL, for his expertise, availability and commitment to education. His continuous support and insightful discussions helped me throughout the pursuit of this degree. Mr. Walter SCHÖN who presided over the jury of the viva, deserves a great deal of thanks.

This work would not have been possible without the implementation of the joint PhD program between the UT-INSA network and the Lebanese University. I am very grateful to Mr. Olivier GAPENNE Dean of the Doctoral School "ED 71: Sciences pour l'ingénieur" at the Université de Technologie de Compiègne, and Mrs. Zeinab SAAD, Dean of the Doctoral School of Science and Technology (EDST) at the Lebanese University.

I thank my colleagues at the Heudiasyc Laboratory and the EDST for their support and encouragement. Special thanks to my beloved and best friends in Lebanon, France, and around the world (across the globe) for their encouragement and great friendship.

Nobody has been more important to me in the pursuit of this project than the members of my family. I would like to express my utmost gratitude to my parents, for their love and unending support.

Contents

List of Figures

List of Tables

Résumé étendu

La classification traditionnelle, dite « mono-label », suppose que chaque individu à classer appartienne à une et une seule classe parmi un ensemble prédéfini de classes supposées mutuellement exclusives. La classification multi-label quant à elle permet d'associer plusieurs classes simultanément à un exemple donné. Dans ce type de problème, les classes ne sont pas nécessairement exclusives [93][111].

La classification multi-label est largement requise par différentes applications actuelles telles que la catégorisation de documents selon leur contexte [16], la classification d'images en fonction de leur contenu sémantique [8][81], la classification de chansons par rapport aux émotions qu'elles évoquent [66], etc.

Plusieurs méthodes ont été proposées pour traiter la problématique de l'apprentissage multi-label [60]. Ces méthodes peuvent être divisées en trois groupes selon la façon dont on traite les données d'apprentissage. Le premier groupe de méthodes transforme le problème d'apprentissage multi-label en un ou plusieurs problèmes d'apprentissage mono-label. Le second groupe se base, quant à lui, sur l'adaptation directe des algorithmes de classification mono-label pour l'apprentissage multi-label. Le troisième groupe résout le problème multi-label par un ensemble des méthodes appartenant aux deux groupes précédents.

Dans l'apprentissage multi-label, l'appartenance d'un individu à une classe apporte de l'information sur l'appartenance de ce même individu à une autre classe. Par exemple, les articles de journaux sont plus susceptibles d'être associés aux classes *science et environnement* qu'aux classes *environnement et sport*. Dans le domaine médical, les personnes obèses sont plus sujettes à l'intolérance au glucose (problème relié au diabète) qu'à la migraine, etc.

L'objectif de cette thèse est la conception de nouvelles techniques pour répondre au problème de classification multi-label. La première partie de ce travail a été consacrée au développement d'une méthode originale basée sur l'approche *RAKEL* (RAndom k-labELsets), qui appartient au dernier groupe de méthodes multi-label. Le principe de cette approche est de générer aléatoirement des sous-ensembles de classes de petite taille par rapport au nombre total des classes et d'apprendre chaque sous-ensemble par un classifieur multi-label. Étant donné un individu $\mathbf{x}$ à classer, la prédiction des classes de $\mathbf{x}$ est déterminée par une stratégie de vote en utilisant un seuil préfixé. La méthode *RAKEL* a l'avantage de tenir compte des corrélations éventuelles entre les différentes classes. En revanche, cette méthode induit une perte d'information puisqu'elle réalise l'apprentissage sur des sous-ensembles de labels. De plus, dans cette méthode, trois paramètres sont à identifier : le nombre de sous-ensembles (ou, d'une manière équivalente, le nombre de classifieurs), la taille de ces sous-ensembles et le seuil utilisé lors de l'application de la stratégie de vote [95].

Un cadre important pour la présentation des connaissances est celui de la théorie des fonctions de croyance, appelé encore théorie de Dempster-Shafer ou théorie de l'évidence [80]. Dans ce formalisme, les connaissances sont représentées par des fonctions de croyance définies sur des cadres de discernement. Soit $\mathcal{Y}$ l'ensemble de toutes les classes possibles pour un problème donné. Dans le cas multi-label, le cadre de discernement, défini comme étant l'ensemble de toutes les hypothèses possibles pour un problème donné, est l'ensemble de toutes les combinaisons possibles de classes, noté $\Omega = 2^{\mathcal{Y}}$. Ainsi, les fonctions de masses sont définies de l'ensemble des parties de Ω vers l'intervalle $[0, 1]$. Avec l'augmentation du nombre de classes, le nombre de valeurs à manipuler croît avec une complexité doublement exponentielle. Pour remédier à ce problème en utilisant la notion des variables multi-valuées, les auteurs dans [25][26] proposent de ne pas considérer l'ensemble 2^{Ω} en entier mais seulement un sous-ensemble clos par intersection et ayant un structure de treillis. Ce nouvel ensemble est suffisant pour modéliser le problème de classification multi-label avec une complexité polynomiale en fonction du nombre de classes. Motivés par cette formulation et afin de réduire la perte d'information causée par l'utilisation de la méthode *RAKEL* (du fait que l'apprentissage de classifieurs est effectué sur un sous-ensemble de labels) tout en tenant compte des corrélations possibles entre les différentes classes, nous avons proposé de conserver la structure de l'approche *RAKEL* et de combiner les sorties de différents classifieurs dans le cadre de la théorie des fonctions de croyance, en utilisant le formalisme de représentation de l'incertitude pour les variables multi-valuées. La méthode développée, baptisée

RAKEL Évidentielle, présente l'intérêt de ne pas nécessiter de fixer un seuil pour appliquer la stratégie de vote, la décision étant prise après combinaison des fonctions de masse par la règle de Dempster. Nous illustrons la méthode *RAKEL Évidentielle* sur des données synthétiques et ainsi que sur des jeux de données communément utilisés pour l'évaluation des méthodes de classification multi-label. Les résultats obtenus montrent l'intérêt et l'efficacité de notre méthode par rapport à la méthode *RAKEL* classique.

À notre connaissance, aucun travail n'avait été effectué sur l'édition ou la purification des bases d'apprentissage pour la classification multi-label. L'une des contributions de cette thèse est l'introduction d'un algorithme simple pour traiter ce problème. En effet, les bases d'apprentissage utilisées dans les applications multi-label peuvent contenir des données corrompues par des erreurs dans les vecteurs d'entrée ou dans leurs étiquettes. Ceci peut avoir des effets néfastes sur les performances des méthodes de classification. L'algorithme que nous présentons est basé sur la méthode des k plus proches voisins (k-ppv) conjointement avec un critère de performance multi-label, appelé le coût de Hamming, qui permet de détecter les exemples mal classés dans l'ensemble d'apprentissage. En se basant sur la valeur de ce critère, les moins bons exemples d'apprentissage sont sélectionnés puis éliminés. Le nouvel ensemble d'apprentissage « purifié » servira pour l'apprentissage des classifieurs multi-label.

Afin de caractériser les performances de notre algorithme de purification, nous l'avons mis en oeuvre avec la méthode des k-ppv crédibiliste (EMLkNN) ainsi qu'avec la méthode des séparateurs à vaste marge pour la classification multi-label (Rank-SVM). Nous avons comparé pour ces deux méthodes sur des données synthétiques et réelles avant et après purification. Les performances ont été évaluées selon différents critères souvent utilisés dans le contexte de la classification multi-label, ainsi que le temps d'apprentissage et la capacité mémoire pour stocker les données. Les résultats obtenus montrent que la purification des données d'apprentissage permet d'améliorer les performances de méthodes de classification multi-label avec une diminution du temps d'exécution ainsi qu'une réduction de l'espace de stockage requis.

Introduction

When you hear a classical piece of music on the car radio, how do you feel? Do you feel happy, sad, amazed, quiet, or angry? Could a computer system predict these emotions? This problem does not seem very complicated, since anyone can express his feelings based on his mood and the memories that this piece of music evokes.

However, there is a huge list of human emotions that we can experience, sometimes simultaneously. In this case, what should be the basis for assigning some of these to this song? Is it the genre, the rhythm, the style or the instrumentation? An approach could be to look through a musical database for a song with similar features. However, how could this database be annotated? Manually or automatically using some digital music collection?

This is an example of the multi-label learning task: to assign one object to one or more classes from a predefined set of classes. Multi-label learning differs in several aspects from traditional single-label learning. In multi-label learning, classes are not necessarily exclusive, which means that the assignment of an instance to a certain class may provide information about the membership of this instance to other classes. For example, newspaper articles are more likely to be assigned with both labels *politics* and *economy*, than they are with both *economy* and *environment*. In the medical domain, patients with diabetes are more susceptible to infections than those having migraine [71][109][111].

Multi-label learning covers a wide range of application areas. For instance, an electronic document can belong to multiple topics: technology, sport, economy. A medical patient may be affected by more than one diseases: diabetes, obesity, high cholesterol, etc. [64]. A piece of music can evoke several emotions: amazed, happy, excited. One protein might have multiple functional labels: metabolism, energy, cellular biogenesis [112].

In order to tackle with such learning tasks, numerous methods have been proposed for building multi-label classifiers. These methods can be categorized into three groups: problem transformation, algorithm adaptation and ensemble methods. Problem transformation methods convert the multi-label learning problem into one or more conventional single-label sub-problems, which are handled separately using traditional existing approaches; a multi-label classifier is then obtained by combining all sub-classifiers. In contrast, algorithm adaptation methods extend some specific learning algorithm to cope with multi-label training datasets directly. The last category refers to methods that use ensembles to make multi-label predictions; their base classifiers belong either to problem transformation or algorithm adaptation methods [60][87].

In multi-label learning, labeled examples are generally annotated by experts. In many applications, there is no ground truth for assigning unambiguously a set of labels to an instance. An expert may be uncertain about the labelling of an instance. Moreover, several experts may have conflicting opinions, which can be confusing when learning from the obtained dataset. For instance, in our emotion prediction example, an expert may decide that a song evokes happiness and relaxation, while another might judge that this music can induce nostalgia. These problems might introduce some uncertainty in the labeling process.

This thesis is about multi-label learning. This task is to learn from a set of previously labeled examples and to build a model that can automatically predict the set of labels for new instances. Our objective is to improve the performances of existing multi-label classifiers.

In our work, we focus on the improvement of a classification method that belongs to the third group of multi-label methods, ensemble methods. This method, called *RAKEL* (RAndom-k-labEL sets), aims at solving the multi-label classification problem while taking into consideration the correlation between labels [95]. It randomly breaks the set of labels into smaller sets and learns a single-label classifier for each subset. To make a decision, the different predictions for each label are aggregated via voting. This method reduces the complexity of multi-label methods belonging to the first category, known as problem transformation methods. However, RAKEL suffers from loss of information as each base classifier only considers a subset of labels. Moreover, in this approach, the user has to identify the number of random label sets, the size of these sets and an adequate threshold in the voting process.

One important framework for the presentation of knowledge is the context of the theory of belief functions, also called evidence theory or Dempster-Shafer theory. In this context, knowledge is expressed in terms of belief functions defined on frames of discernment. Let $\mathcal{Y}$ be the set of all possible classes in a multi-label learning problem. The frame of discernment, defined as the set of mutually exclusive and exhaustive hypotheses about some problem domain, is the set of all possible combinations of classes; it is denoted by $\Omega = 2^{\mathcal{Y}}$. Mass functions are then defined from the power set of Ω to $[0, 1]$. This approach often implies working in a space of very high cardinality, when the number Q of classes is large. To alleviate this problem, the authors in [25][26] propose an approach based on the definition of a restricted family of subsets of 2^{Ω} that is closed under intersection and has a lattice structure. This framework is able to express rich information about the problem of multi-label classification with only a moderate complexity. Based on this work and in order to alleviate the loss of information inherent in the *RAKEL* method while accounting for label correlation in a more efficient way, we propose to retain the basic principle of the *RAKEL* approach while handling uncertainties in multi-label problems by combining the different outputs in the framework of Dempster-Shafer theory.

In general, multi-label classifiers are learnt from complete training sets where each instance is correctly described by its own features and it is associated with a precise set of labels. However, collecting such high quality information may be impractical due to the size of data and the cost of human experts. This is why real-world datasets are never as good as we would like them to be and can contain various kinds of errors, either random or systematic. Therefore, the success of machine learning algorithms depend very much on the learner's ability to handle imprecisions and uncertainties in data labeling and to deal with erroneous data.

One intuitive way to alleviate problems incurred by previously labeled instances, as already mentioned, consists of identifying dubious instances that may be erroneous or mislabeled and eliminating them from the training set. This approach, referred to as *prototype selection*, provides an appropriate reduced subset of prototypes that leads to improving the performances of some classifiers and reducing the conceptual costs of learning methods.

Summary of contributions

In this thesis we propose two original methods that address the multi-label classification problem. These two methods attempt to exploit correlation between labels and to handle imprecise and uncertain information while taking into account specificities of multi-labeled data.

Our first contribution tackles the problem of learning using ensemble methods. The developed method is called *Evidential RAKEL* (published in [48] and [49]). It is an extended version of the classical *RAKEL* based on the voting process. These two methods apply different classifiers on several subsets of labels selecting from the whole set of possible labels. However, *Evidential RAKEL* converts each output from a base classifier into a mass function. These mass functions are combined and a final decision is made. To evaluate our method, we applied *Evidential RAKEL* to synthetic and real-world datasets. Our experimental results show that *Evidential RAKEL* performs better than traditional classifiers such as classical *RAKEL* and Ensemble of Multi-Label classifiers (EML).

Our second contribution addresses the problem of prototype selection in the framework of multi-label learning. To the best of our knowledge, this problem had not seen addressed so far. In the literature on single-label learning, a large number of methods has been proposed for prototype selection. Many of these methods tend to edit training datasets to keep the most reliable instances. Most editing algorithms are based on the k-nearest neighbor rule, and the edition is done by comparing the predicted and the ground truth classes for each training instance [36]. In contrast, in multi-label learning, each training example can be labeled by several classes and a classifier can correctly predict some of them and induce misclassifications for others. To select the erroneous or mislabeled data, we propose a new evaluation measure inspired by the well-known metric, Hamming loss, which assesses the predictive performance by using the average difference between the truth and the predicted sets of labels [93].

Our editing algorithm has been applied to a wide variety of data sources (published in [50]). Two multi-labeled classifiers are learnt from the obtained edited data. These classifiers achieve better performance according to several measures, when learning from edited data than they do when learning from the initial datasets. An analysis of time complexity shows that learning from edited data makes it possible to reduce the running time for some classifiers.

Organization of the thesis

The thesis is structured as follows. Chapter 1 introduces the framework in which the research is placed. It briefly presents an overview of existing multi-label methods, application domains and performance criteria. Chapter 2 outlines the general background on belief functions theory and its application to multi-label learning. In Chapter 3, we formalize our learning algorithm based on the RAKEL approach in the belief function framework and presents the experimental results. In Chapter 4, we propose a novel editing method in the field of multi-label learning. This chapter begins by a review of single-label editing methods, introduces our editing algorithm and discusses some experimental results. General conclusions and future works conclude this manuscript.

Chapter 1

Multi-label learning

1.1 Introduction

Machine learning is usually defined as the field of study that gives computers the ability to learn without being explicitly programmed. This learning is possible thanks to previous observed circumstances or to computers models that describe problems from a given dataset. Machine learning techniques have traditionally been divided into three categories: supervised, unsupervised and semi-supervised learning.

In supervised learning [53], the system receives a dataset with different parameter values and different decisions (for example classes for a classification task), and the goal is to find the best model which automatically maps an input example to the correct output class. Supervised learning methods are particularly applicable to classification and regression problems. In the case of classification, the outputs take discrete values from a finite set known as the set of classes (or labels). In the case of regression, the outputs are numerical.

In unsupervised learning, elements of the dataset are not labeled. Unsupervised learning methods look for similarities within the dataset without any need for prior labelling of the data. Typical unsupervised tasks include clustering that groups similar items into clusters, and dimensionality reduction which maps a set of high dimensional input instances into a lower dimensional space while preserving certain properties of the dataset [43].

Semi-supervised learning is halfway between supervised and unsupervised learning [14]. In semi-supervised learning, the key idea is to use a large amount of unlabeled data with the labeled data to produce classifiers. This thesis focuses on the supervised learning problem for classification.

This chapter gives a definition and summary of works related to supervised classification and especially to multi-label learning. It is organized as follows. Section 1.2 discusses the different types of classification for supervised learning. Section 1.3 explains the different multi-label learning approaches. A collection of real-world applications requiring multi-label learning is described in Section 1.4. We dedicate some discussions to statistics describing multi-labeled data in Section 1.5. Finally, we provide a set of evaluation metrics for predictive performance in Section 1.6, and we end with the conclusion in Section 1.7.

1.2 Classification problems

In the supervised learning domain, there are two different phases: the training phase and the testing phase. In the training phase, the machine learning method learns a classifier from training data in order to produce a model that maps the attributes of the training examples to the corresponding target class (class or classes). In the testing phase, the computer model is used to label unseen instances in order to evaluate the classifier performance (see Figure 1.1).

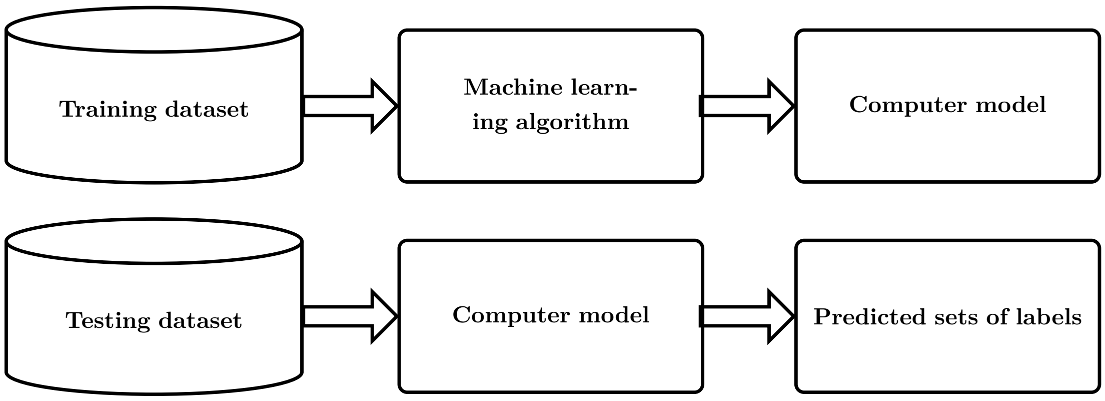


Figure 1.1: Overview of machine learning system

Let $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ be a set of n training instances generated according to some distribution, and let $\mathbb{X}$ be the domain of instances. Let $\mathcal{Y}$ be the set of target classes. The goal is to find a classifier that predicts the class of each unseen instance.

Hereafter, we will discuss different supervised learning tasks. Machine learning algorithms can be categorized into different groups according to the partition of instances and their target categories.

Single-label classification

Single-label (or traditional) classification is the task of automatically assigning a single class label to each input instance, where a classifier learns to associate to every unseen element its most probable class or category. In general, single-label classification problems can be divided into two main groups: binary and multi-class problems.

The binary classification problem is the simplest case, where the set of classes is restricted to two. In this context, we distinguish between the positive ($+1$) and the negative (-1) class, and we denote by positive instances, those having the positive class as a true label and vice versa. A simple example of the binary-class problems is when a women goes to the doctor to find out if she is pregnant. The result of the test may be either positive or negative. Many learning algorithms for binary classification are presented in the literature, such as support vector machines [85] and kernel Fisher discriminants [34].

When the number of classes is greater than two, the learning problem is called multi-class classification. It is assumed that target classes are disjoint and exclusive, in the sense that every element belongs to precisely one class. Multi-class classification is encountered in various fields; for instance, an individual has one blood type among a set of four types: A, B, AB or O. Generally, multi-class classification problems can be solved in two ways; using either direct multi-class learning algorithms or decomposition-based approaches that combine several binary-class classifiers [68]. The former category includes methods such as: neural networks [7], decision trees [67], k-nearest neighbors [6] and naive Bayes classifiers [2].

The one-class classification problem is the one where an instance can belong to only and only one class from a set of predefined classes. One-class classification differs from the binary and multi-class classification by the fact that the training set contains only objects of the target class and no information is available about the other classes. The task is to define a boundary that encloses the target objects and minimizes the chance of accepting outlier objects [51]. This type of classification is widely used in the area of machine diagnostics: the positive class is then defined by measurements corresponding to the nominal state of the machine.

Multi-label classification

Multi-label classification is the task of assigning to an input instance multiple classes simultaneously from a set of disjoint classes; the classes are then no longer mutually exclusive. In this context, we often use the term "label" instead of "class". Each instance is associated with a set of relevant labels. The remaining labels are considered as irrelevant.

Contrary to single-label classification, the multi-label task is influenced by intrinsic latent correlations between labels, in the sense that the membership of an instance to a class can be helpful to predict its set of labels [111]. For example, a patient with a high blood pressure is more likely to develop heart disease than an other person, but less likely to develop a muscular dystrophy.

Multi-label learning has important applications in many real-world problems like text categorization, scene classification, video annotation and bioinformatics, where the task is to predict for an example a set of labels whose size is a priori unknown [8][16][66][81]. It's clear that the single label problem can be considered as a special case of the multi-label learning problem.

Multi-label learning is related to multi-label ranking. Multi-label ranking is the task of learning a mapping from instances to rankings over the set of labels, such that the relevant labels are ranked higher than the irrelevant ones [31][57][60].

Multi-instance learning

Multi-instance or multiple-instance (MI) learning refers to a learning scenario where each example is represented by many alternative feature vectors (instances) instead of one single instance [29]. In this context, the set of instances called a bag has a class label, but the instances themselves are not explicitly labeled. A bag is labeled as positive if it contains at least one positive instance; otherwise it is labeled as negative.

The oldest multi-instance application is the drug activity prediction. A molecule treated as a bag of conformers, is biologically active if and only if one of its conformers is responsible for the observed bioactivity, and a molecule is inactive if none of its conformers is responsible for the observed activity [33].

Algorithms proposed in this area can be divided into two groups. The first group of methods handles directly the multi-instance problems. An example in this category is the *three axis-parallel rectangles algorithm* which attempts to look for an axis-parallel rectangle in the feature space to represent the target concept [29]. The second group tries to adapt existing learning algorithms to solve the multi-instance problem. This group includes the Bayesian-kNN [103] and ID3-MI [17] methods, among others.

The most general variant of multi-instance learning is multi-instance multi-label (MIML) learning, where each example can be described by multiple instances and associated with multiple labels. For example, in text categorization, a document can be represented by a bag of instances (several sections or paragraphs), and may belong to several topics such as sports and politics. To solve such problems, existing methods include the MIMLBoost algorithm [116], the MIML support vector machines [116] and the MIML k-nearest neighbors techniques [114].

Hierarchical classification

Hierarchical classification refers to the situation where categories have a hierarchical structure. Recently, researchers investigated the use of hierarchies to multi-label classification domain [12][83][100]. The corresponding learning task is referred to as hierarchical multi-label classification. Each input element is assigned to more than one class and the hierarchy of classes is expressed by generalization trees or directed acyclic graphs [72][100].

Hierarchical classification is required by many modern applications such as collections of web pages, biological functions of genes or Text categorization. For example, in web page categorization, two top-level categories: Business and Computers can be divided into multiple subcategories: Business/Jobs, Business/Inverting, Computers/Internet, Computers/Hardware and Computers/Software.

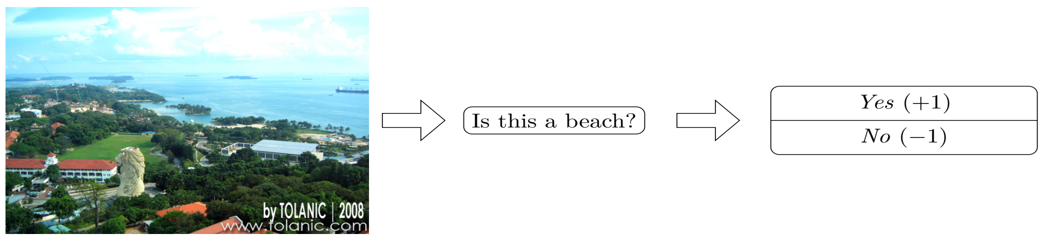
To deal with hierarchical learning problems, different algorithms have been proposed in the literature [83][98]. These algorithms can be categorized into three groups: flat classification, local classification (or Top-Down) and global classification (or Big-Bang) approaches. In the flat classification approaches, the hierarchical problem is transformed to a flat classification problem where the class hierarchy is ignored and the classification problem handles only the leaf-node classes. Each leaf-node is treated by an ordinary classifier (multi-class or multi-label). Local classification approaches apply one or more classifiers to each class node or class level of the hierarchy and each classifier works as a flat classifier at that level, while global classification approaches construct one global classification model for all classes created in a single run of the algorithm.

Example

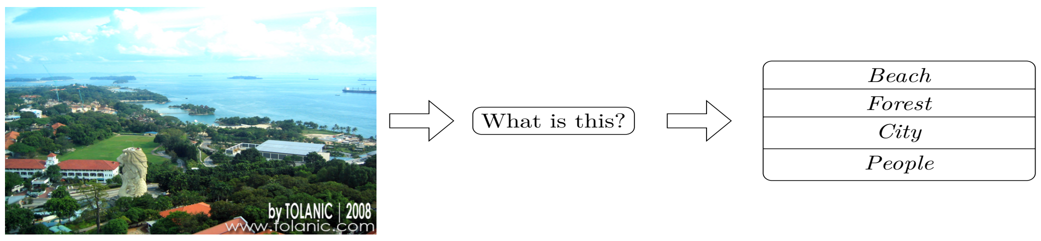
Based on the observed features for a given example, and the number of predefined classes as well as the mutual relation between these classes, the follow example can illustrate the different problems outlined above. Figure 1.2 gives an example of all previous types of classification for an image taken from the cable car in Singapore to the Sentosa island. Figure 1.2(a) shows the results on the binary classification by making a "Yes" decision based on the "beach" viewed in this picture. Figure 1.2(b) shows the multi-class classification by associating one and only one class to the image. In Figure 1.2(c), the image is considered as multi-labeled and three classes are involved, say "Forest", "City", and "Beach". In Figure 1.2(d), the image is divided into multiple regions in which we can see different objects like "Sea", "Trees", "Hotels", "Roads", "Grassland", "Building" and "Resort". According to these regions, multi-instance multi-label learning assigns the output to three classes, "Forest", "City", and "Beach". Based on hierarchical classification, different categories are organized in a hierarchical tree-like structure as illustrated in figure 1.2(e). Each category has a certain number of subcategories describing the Sentosa island. In this island, we can find a variety of attractions including the "Universal Studios of Singapore", three artificial beaches and a forest containing "Fauna" and "Flora".

1.3 An overview of approaches for multi-label learning

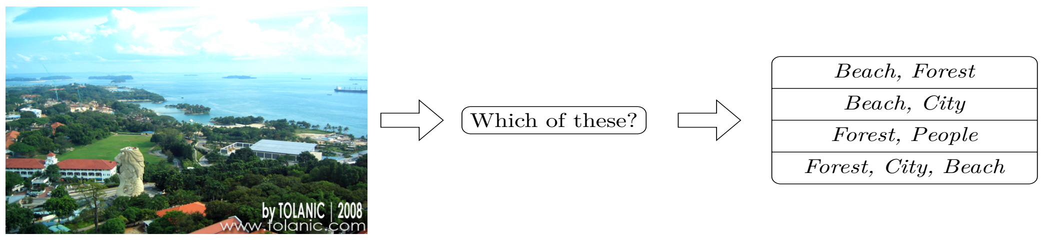
Let $\mathbb{X}$ denote an instance space, and let $\mathcal{Y} = \{\omega_1, \ldots, \omega_Q\}$ be a finite set of labels. Let $\mathcal{D} = \{(\mathbf{x}_1, Y_1), \ldots, (\mathbf{x}_n, Y_n)\}$ denote a dataset composed of n multi-labeled objects $(\mathbf{x}_i, Y_i)$, $\mathbf{x}_i \in \mathbb{X}$ and $Y_i \subseteq \mathcal{Y}$, where each instance is independent and identically distributed (i.i.d.) drawn from an unknown distribution. The goal of multi-label learning is to build a multi-label classifier $\mathcal{H}$ that maps an instance $\mathbf{x}$ to its associated set of



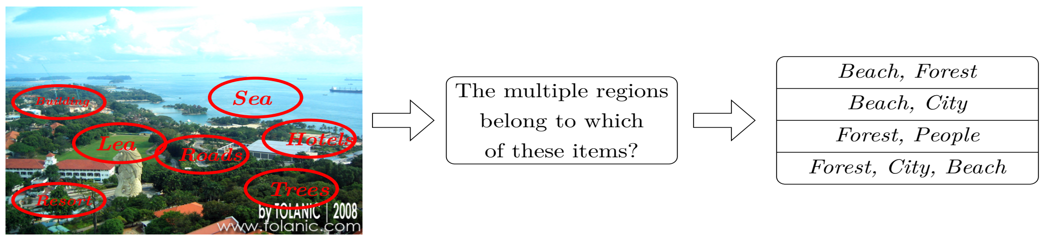
(a) Binary Classification



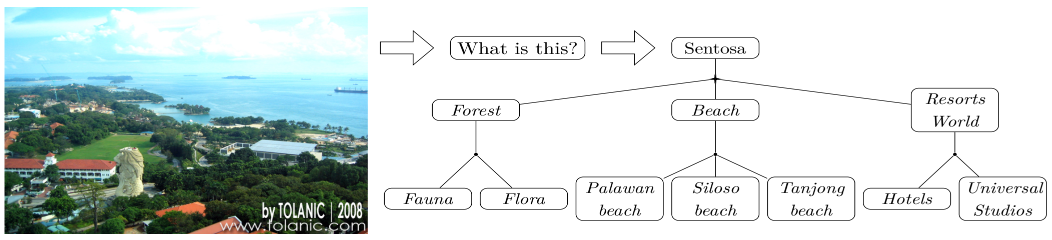
(b) Multi-class Classification



(c) Multi-label Classification



(d) MIML Classification



(e) Hierarchical Classification

Figure 1.2: Different types of classification problems

labels Y and optimizes some evaluation metrics. Here, the set of all subsets of $\mathcal{Y}$ is the power set of $\mathcal{Y}$ denoted by $2^{\mathcal{Y}}$.

Numerous methods have been proposed in the literature to deal with multi-label learning problems. Existing algorithms can be grouped into three categories as proposed in [60]: problem transformation approaches, problem adaptation algorithms and ensemble methods. The first category divides the multi-label problem into one or more conventional single-label problems. The second category generalizes single-label algorithms to cope with multi-labeled data directly. Finally, the third category incorporates the merits of these two previous approaches.

In the following paragraph, we will explain in more details each of these categories and describe the main characteristics of corresponding methods. Figure 1.3 shows the different categories and associated methods.

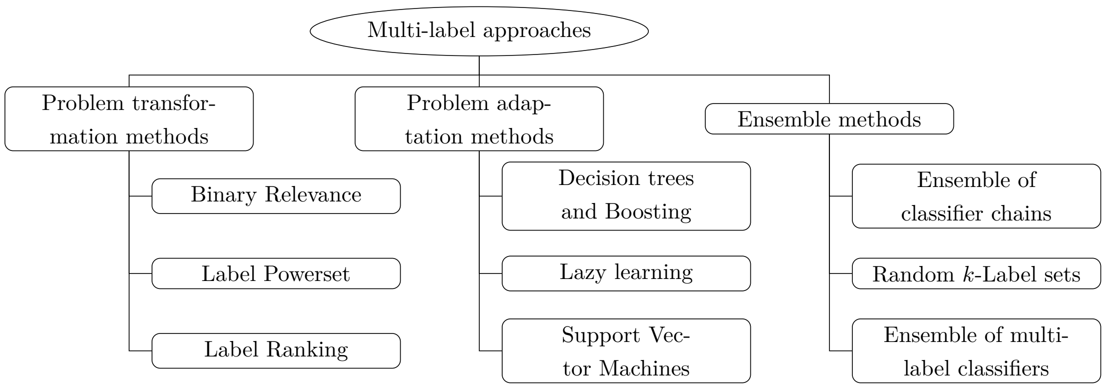

Multi-label approaches
Problem transformation methods
Problem adaptation methods
Ensemble methods
Binary Relevance
Label Powerset
Label Ranking
Decision trees and Boosting
Lazy learning
Support Vector Machines
Ensemble of classifier chains
Random k-Label sets
Ensemble of multi-label classifiers

Figure 1.3: Multi-label learning approaches [60]

1.3.1 Problem transformation methods

The simplest strategy in multi-label learning is the problem transformation approaches which can be used with any learning algorithm. In these methods, the multi-label classification problem is transformed into one or more single-label classification problems. The solutions of these problems are then combined to solve the original task of multi-label learning. Problem transformation methods include three principal approaches: binary relevance, label powerset and label ranking.

Binary relevance

The Binary Relevance (BR) approach, also known as the one-against-all strategy, divides the multi-label learning problem with Q possible classes into Q single-label classification problems which can be solved by training Q binary classifiers $(\mathrm{h}_1, \ldots, \mathrm{h}_Q)$ [92]. Each q classifier ($q \in \{1, \ldots, Q\}$) is trained on the original dataset, and aims at determining the relevance of its particular label for a given instance. When classifying a new instance $\mathbf{x}$, BR outputs the union of labels that are positively predicted by the binary classifiers. The multi-label classifier is then determined by: $\mathcal{H}(\mathbf{x}) = \{\omega_q \in \mathcal{Y} | \mathrm{h}_q(\mathbf{x}) = 1\}$. Many well-known *classical* classification methods are generalized to handle the multi-label problem using the BR approach, e.g., decision trees, SVM and k-NN. The BR approach is simple to implement, and its complexity is linear with the number of possible labels. However, BR ignores correlation between labels by treating each label independently. To deal with the negative aspects of BR, the Classifier Chain (CC) introduced in [71] involves Q binary classifiers linked along a chain. The *Label Powerset* approach presented in the next Section constitutes also one of the alternatives to deal with these negative aspects of the BR approach.

Label Powerset

Given a training dataset $\mathcal{D}$ with n instances, the Label Powerset (LP) approach considers each unique set of labels in $\mathcal{D}$ as a single label and then trains a single-label classifier. The number of classes is upper bounded by $\min(2^Q, n)$. The complexity of LP relies on the complexity of the single-label classifier with respect to the number of classes. For a new instance, the LP approach outputs the most probable class which is a set of labels in the original multi-label representation. LP has the advantage of taking into account label correlations. Moreover, a negative aspect of this approach is that it may lead to imbalanced datasets with a large number of classes associated with few examples. This issue is resolved by considering only the label combinations frequently found in $\mathcal{D}$ as class values for the single-label classifier. This new approach is called pruned transformation method [69].

Label Ranking

The Label Ranking (LR) approach learns a mapping from instances to rankings over all possible labels. Given a relevance ranking over labels for an instance, single-label classification selects the most relevant label (or class) for this instance. In contrast, in the multi-label case, the *topmost* labels, and not only the top label, are related to that instance. To output the label set for an unseen instance, the LR approach splits the ordered label set into relevant and irrelevant labels. The goal of multi-label learning is thus to find a scoring function $f : \mathbb{X} \times \mathcal{Y} \longrightarrow \mathbb{R}$ that assigns a score or value to each couple $(\mathbf{x}, \omega) \in \mathbb{X} \times \mathcal{Y}$. To determine classes that should be assigned to a particular instance, a threshold value is introduced. The output of the multi-label classifier $\mathcal{H}$ is thus: $\mathcal{H}(\mathbf{x}) = \{\omega \in \mathcal{Y} | f(\mathbf{x}, \omega) \geq t\}$, where t is a threshold, usually fixed to a constant value. The LR approach does not explicitly model label correlations.

1.3.2 Problem adaptation algorithms

Problem adaptation methods customize traditional machine learning algorithms in order to handle multi-label concepts directly. These methods have the advantage of focusing on one specific algorithm. Another advantage is that these methods use the whole training dataset (instances and labels) at once to train a multi-label classifier. In general, the performances of these algorithms are better in difficult real-world problems than those of problem transformation methods, at the cost of higher complexity. Several adaptations of traditional learning algorithms have been proposed in the literature, some of them extend all labels and instances simultaneously, like decision trees [18] and the multi-label support vector machines [31], while others consider each class separately, like the multi-label k-nearest neighbors method (MLkNN) [115]. These methods are briefly reviewed below.

Decision trees and Boosting

Decision trees are among the most popular methods for classification in machine learning. Starting with all training instances, a decision tree creates a predictive model having a tree structure, which can be viewed as a partitioning of the training dataset. Nodes in the tree represent attributes that are connected to branches which lead to child nodes (partitions). For an unseen instance, the target class is predicted by following the path of nodes and branches from the root node to the terminal leaf.

Many decision trees algorithms have been extended to multi-label classification domain. In [18], a C4.5 multi-label algorithm is proposed by modifying the formula of entropy. A large number of leaves is generated for all combinations of different labels. This method can handle multiple labels on several levels of the hierarchy and assign a larger cost to misclassifications higher up in the hierarchy. It was evaluated first on multi-labeled data coming from functional genomics.

Boosting was applied to multi-label learning and especially to text categorization [77]. Boosting-based methods include two slightly different versions of the ensemble learning method AdaBoost (AdaBoost.MH and AdaBoost.MR). The former extension aims at predicting the set of all correct labels for a query instance, while the latter is designed to find a classifier that ranks the labels so that the correct labels will receive the highest ranks. AdaBoost.MH is combined with the alternating decision trees algorithm to produce the Adapted Decision Tree Boosting, ADTBoost.MH in [23].

Support Vector Machines

Support Vector Machines (SVM) have widely been used with problem transformation methods, especially with the BR approach, to resolve the multi-label learning problem [39][104][109]. However, to the best of our knowledge, only one method has been proposed in the literature in order to extend the SVM algorithm to handle the multi-label problem directly.

Rank-SVM defines a linear model based on a ranking system combined with a label set size predictor [31]. In the original input space, Q linear discriminative functions are defined as, $(q \in \{1, \ldots, Q\})$, $f_q(\mathbf{x}) = \langle w_q, \mathbf{x} \rangle + b_q$, where w_q and b_q denote the weight vectors and bias terms. This model is decomposed into two parts. The former follows a ranking system, which orders the labels according to their output values. The latter defines a set size predictor $(t(\mathbf{x}))$ which can serve as a threshold value to differentiate the relevant labels from the others, $(f_q(\mathbf{x}) > t(\mathbf{x}))$.

This procedure aims at minimizing the ranking loss, defined as the average fraction of label pairs that are not correctly ordered, while holding a large margin. More details on this method are given in Chapter 4. In [110], a zero label is added to find a natural zero point to detect relevant labels. The new model simplifies the original form of Rank-SVM and leads to a novel quadratic problem with special linear constraints.

Lazy learning

Several lazy learning algorithms have been adapted to multi-label tasks. In [115], the authors proposed a multi-label classification method based on the k-nearest neighbors (k-NN) algorithm, named MLkNN. In this method, maximum a posteriori estimation is used to determine the proper set of labels to be assigned to a test instance $\mathbf{x}$, according to statistical information extracted from the labeling of the nearest neighbors. For each ω_q in $\mathcal{Y}$, the numbers of neighboring instances belonging to each possible class are used in order to compute the posterior probabilities that $\mathbf{x}$ belongs or does not belong to ω_q. Depending on which of these probabilities is greater, we decide to assign or not the class ω_q to test instance $\mathbf{x}$. The decision is made independently for each label.

An improvement on the MLkNN method, called Dependent Multi-Label k-nearest neighbor (DMLkNN), was introduced in [111]. This method generalizes the MLkNN algorithm by relaxing the assumption of label independence. The correlation between labels is exploited when computing the two aforementioned probabilities. When computing the posterior probabilities for ω_q, the frequency of co-occurrence of a label ω_r in the label sets of the neighboring instances affects the membership of $\mathbf{x}$ to class ω_q. The DMLkNN method takes into account these dependencies between labels.

Two extensions of the k-NN method for the multi-label classification are presented in [59]. These two versions, applied to document categorization, consider the co-occurrences in documents of multiple categories. Given a new instance, the labels associated with the k-nearest neighbors are retrieved, and a counter corresponding to each label is increased if that label exists in each of these neighbors. The test instance is thus classified with the top N weighted labels (represented by the largest counts). N is chosen based on the number of labels of the instance. These methods are unsuitable when the number of labels is not identified for a new instance [93].

A multi-label learning algorithm following the paradigm of associative classification through associative rules is proposed in [89]. The multi-class multi-label associative classification (MMAC) is divided in three modules: rules generation, recursive learning and classification. In [99], the idea of associative rules is combined with lazy learning which iteratively exploits dependencies between labels.

1.3.3 Ensemble methods

Ensemble methods incorporate problem transformation and problem adaptation classifiers. Many of these methods can be aggregated to output a new classifier for multi-label learning. The aggregated algorithms can be homogeneous in the sense that one can use the same algorithm for each classifier, or heterogeneous, when various algorithms contribute in building of the final classifier. Ensemble methods may alleviate disadvantages of one base-classifier by adding ensemble of classifiers. Several ensemble methods have been proposed, among them: ensemble of classifier chains [71], random k-label sets [95] and ensemble of multi-label classifiers [87].

Ensemble of classifier chains

Ensemble of Classifier Chains (ECC) use Classifier Chains (CC) as base classifiers [71]. ECC was proposed to relieve the effect of classifier order in CC, by training an ensemble of CC classifiers, $C_1, \ldots, C_m$. Each C_k can be trained with a random chain ordering on a random subset of $\mathcal{D}$. Given an unseen instance, predictions from different classifiers are gathered and combined for each label so that each label receives a number of votes. To output the final multi-label set, a threshold is used to select the most relevant labels. This ensemble method can use any multi-label problem transformation method as base classifier. In general, ECC outperforms BR and CC while maintaining an acceptable computational complexity.

Random k-labEL sets

The RAndom k-labEl sets (RAKEL) method is related to problem transformation approaches and especially to the LP approach [95]. RAKEL retrieves the problem of LP presented by the large number of labels with few examples per class. This method draws a random subset of size k from all labels and trains a LP-based multi-label classifier for each of the label sets. Given a new instance, the decision of all LP-classifiers are combined using a simple voting process to determine the final set of labels. RAKEL has a number of parameters that should be optimized to get near-optimal performance. This can be difficult when the number of training examples is insufficient. This method will be described in further detail in Chapter 3.

Ensemble of multi-label classifiers

To improve the performance of multi-label classifiers and to address the imbalance problem (very few instances for some labels), the authors in [87] proposed Ensemble of Multi-Label classifiers (EML). EML use heterogeneous ensembles of multi-label learners, which consist of a set of individually trained classifiers: $\mathrm{h}_1, \ldots, \mathrm{h}_q$, each multi-label classifier h_k belongs to different adaptation group. For a test instance $\mathbf{x}$, each individual classifier h_k produces a Q-dimensional vector $P_k = [p_{1k}, \ldots, p_{Qk}]$. Each p_{ik} represents the probability that the membership of instance $\mathbf{x}$ to class ω_i has been correctly assigned by classifier k. The outputs of these q classifiers are aggregated by using different combination techniques based on averaging and weighted voting methods. The results show that these approaches provide significant improvements by tackling the problems of class imbalance and label correlation.

1.4 Applications

Perhaps the oldest application of multi-label classification is *text categorization*, where the task is to assign predefined categories to free text documents. Text categorization applications may include controlled vocabulary indexing (e.g., sorted search results by Google and Yahoo), documents filtering (e.g., email classification between spam and serious mails), word sense disambiguation (e.g., cold = disease, temperature sensation or environmental condition), web search and information security [55][79]. In most applications, text categorization requires multi-label classification. For example, in medical diagnosis, a document report including some symptoms can describe a category of diseases like: high blood pressure, high cholesterol and heart disease.

Automatic music classification is a problem in the field of Music Information Retrieval. Due to advances in technologies such as data compression and network transmission, digital music collections have grown in volume, and the need for automatic retrieval, classification and organization are becoming more and more demanding by the user [74]. Digital music collections are based on textual meta-data, like artist, genre, emotion, rhythm and timbre [117]. Due to the fact that a music piece can belong to an unrestricted set of musical genres, like blues rock and Latin jazz, music genre classification supports multi-label learning algorithms [15][58].

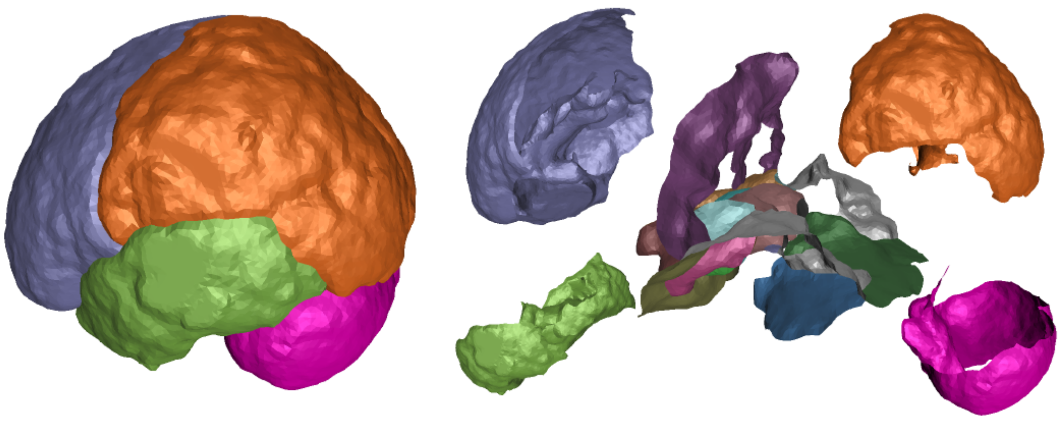


Figure 1.4: MNI structural atlas of the adult human brain [28]

With the huge amount of medical images produced, *automatic image annotation* becomes an important task to improve analysis for patient diagnosis, surgical planning, therapy and medical reference. These 3D images can come from different modalities like Magnetic Resonance Imagery (MRI), Nuclear Medicine Imagery (NMI), ultrasound Imagery (USI) or Computed Tomography Imagery (CTI) [4][62]. Each image is represented by a grid of scalar values. Multi-label learning is required in medical image analysis where each grid is labeled by a region representing structural elements of the scanned object such as bone, muscle or organs [28]. Figure 1.4 shows different regions of the brain for an adult as viewed by the MNI (Montreal Neurological Institute for neuroscience).

Another applications of multi-label classification can be found in bioinformatics where genes and proteins are annotated with their functional roles [47][78][112]. Due to annotation, a gene has several functions of which the biological function is intrinsically a multi-label classification problem. For example, a protein can have enzyme regulator activity with catalytic domain and kinase activity. Figure 1.5 gives an overview of protein-encoding gene functions, where each protein can have some of these activities at the same time.

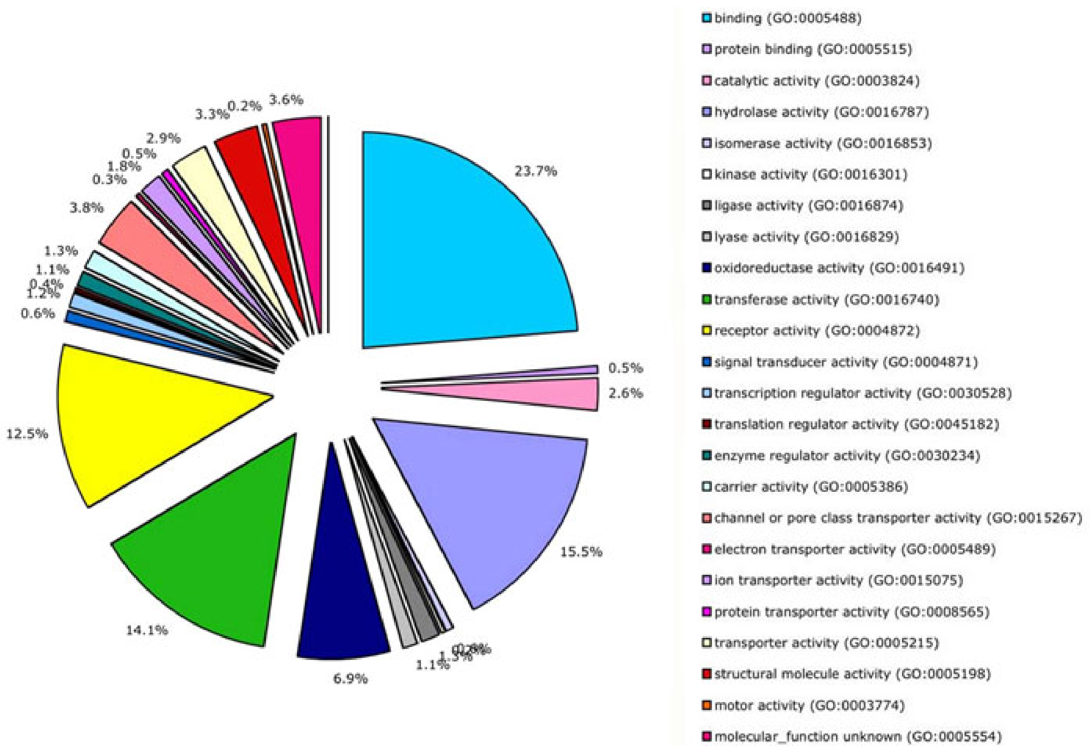


Figure 1.5: Overview of protein-encoding gene functions, as summarized by InterPro families and their mapping to molecular function terms. [2]

1.5 Multi-labeled dataset statistics

Given a multi-labeled dataset $\mathcal{D} = \{(\mathbf{x}_i, Y_i), i = 1, \ldots, n\}$ with $\mathbf{x}_i \in \mathbb{X}$ and $Y_i \subseteq \mathcal{Y}$, this dataset can be measured by the number of instances (n), the number of attributes in the input space, and the number of labels (Q). In the following, we review some statistics about the multi-labeled dataset $\mathcal{D}$ [93].

1. The *Label Cardinality* (LCard) of $\mathcal{D}$ is the average number of labels per instance. Label cardinality is calculated as

$$LCard(\mathcal{D}) = \frac{1}{n} \sum_{i=1}^{n} |Y_i| \tag{1.1}$$

2. The *Label Density* (LDen) of $\mathcal{D}$ is defined as the average number of labels per

2. `http://www.wormbook.org/chapters/www_genomclassprot/genomclassprot.html`

instance divided by the total number of labels Q. Label density is calculated as:

$$LDen(\mathcal{D}) = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i|}{Q} \tag{1.2}$$

Both metrics indicate the number of alternative labels that characterize the examples of a multi-labeled dataset. Label cardinality is independent of the total number of labels in the classification problem, while label density takes into consideration the total number of labels. Two datasets with the same label cardinality but with different label densities may present different properties that influence the performance of the multi-label classification methods.

3. The *Distinct Label sets* (DL) counts the number of label sets that are unique across the total number of examples. Distinct label sets is given by:

$$DL(\mathcal{D}) = |\{Y_i \subseteq \mathcal{Y} | \exists\ \mathbf{x}_i \in \mathbb{X}\ :\ (\mathbf{x}_i, Y_i) \in \mathcal{D}\}| \tag{1.3}$$

This measure gives an idea of the regularity of the labeling scheme.

1.6 Evaluation metrics

Performance evaluation for multi-label learning systems differs from that of single-label classification. Let $\mathcal{H} : \mathbb{X} \rightarrow 2^{\mathcal{Y}}$ be a multi-label classifier that assigns a predicted label subset of $\mathcal{Y} = \{\omega_1, \ldots, \omega_Q\}$ to each instance $\mathbf{x} \in \mathbb{X}$, and let $f : \mathbb{X} \times \mathcal{Y} \rightarrow [0, 1]$ be the corresponding scoring function which gives a score for each label ω_q which in turn is interpreted as the probability that ω_q is relevant. The function $f(.,.)$ can be transformed to a ranking function $rank_f(.,.)$ which maps the outputs of $f(\mathbf{x}, \omega)$ for any $\omega \in \mathcal{Y}$ to $\{1, 2, \ldots, Q\}$ so that $f(\mathbf{x}_i, \omega_q) > f(\mathbf{x}_i, \omega_r)$ implies that $rank_f(\mathbf{x}_i, \omega_q) < rank_f(\mathbf{x}_i, \omega_r)$, (the most relevant label, receives the highest rank (1), while the least relevant one, receives the lowest rank (Q)) [94].

Given a set $\mathcal{S} = \{(\mathbf{x}_1, Y_1), \ldots, (\mathbf{x}_m, Y_m)\}$ of m test examples, the evaluation metrics of multi-label learning systems are divided into two groups: *prediction-based* and *ranking-based* metrics. *Prediction-based* measures are calculated based on the average difference of the actual and the predicted set of labels over all test examples. *Ranking-based* metrics evaluate the label ranking quality depending on the scoring function $f(.,.)$.

1.6.1 Prediction-based measures

Hamming loss: The hamming loss metric for the set of labels is defined as the fraction of labels whose relevance is incorrectly predicted:

$$\mathcal{H}Loss(\mathcal{H},\mathcal{S}) = \frac{1}{m}\sum_{i=1}^{m}\frac{|Y_i \triangle \widehat{Y}_i|}{Q}, \tag{1.4}$$

where $\triangle$ denotes the symmetric difference between two sets.

Accuracy: The accuracy metric gives an average degree of similarity between the predicted and the ground truth label sets:

$$\mathcal{A}ccuracy(\mathcal{H},\mathcal{S}) = \frac{1}{m}\sum_{i=1}^{m}\frac{|Y_i \cap \widehat{Y}_i|}{|Y_i \cup \widehat{Y}_i|}. \tag{1.5}$$

Precision: The precision metric computes the proportion of true positive predictions:

$$\mathcal{P}recision(\mathcal{H},\mathcal{S}) = \frac{1}{m}\sum_{i=1}^{m}\frac{|Y_i \cap \widehat{Y}_i|}{|\widehat{Y}_i|}. \tag{1.6}$$

Recall: This metric estimates the proportion of true labels that have been predicted as positives:

$$\mathcal{R}ecall(\mathcal{H},\mathcal{S}) = \frac{1}{m}\sum_{i=1}^{m}\frac{|Y_i \cap \widehat{Y}_i|}{|Y_i|}. \tag{1.7}$$

F1-measure: $F1$ measure is defined as the harmonic mean of precision and recall. It is calculated as:

$$\mathcal{F}1(\mathcal{H},\mathcal{S}) = \frac{1}{m}\sum_{i=1}^{m}\frac{2|Y_i \cap \widehat{Y}_i|}{|Y_i| + |\widehat{Y}_i|}. \tag{1.8}$$

Note that the smaller the value of the Hamming loss, the better the performance. For the other metrics, higher values correspond to better classification quality.

1.6.2 Ranking-based measures

One-error: This metric computes how many times the top-ranked label is not in the true set of labels of the instance, and it ignores the relevancy of all other labels.

$$OErr(f,\mathcal{S}) = \frac{1}{m}\sum_{i=1}^{m}\langle[\arg\max_{\omega \in Y} f(\mathbf{x}_i,\omega)] \notin Y_i\rangle, \tag{1.9}$$

where for any proposition H, $\langle H \rangle$ equals to 1 if H holds and 0 otherwise. Note that, for single-label classification problems, the One Error is identical to ordinary classification error.

Coverage: Coverage computes the average of how far we need to move down the ranked label list in order to cover all the labels assigned to a test instance.

$$\mathcal{C}ov(f, \mathcal{S}) = \frac{1}{m} \sum_{i=1}^{m} \max_{\omega \in Y_i} rank_f(\mathbf{x}_i, \omega) - 1. \tag{1.10}$$

Ranking loss: This metric computes the number of times that an incorrect label is ranked higher than a correct label.

$$\mathcal{R}Loss(f, \mathcal{S}) = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{|Y_i||\overline{Y}_i|} |(\omega_q, \omega_r) \in Y_i \times \overline{Y}_i \backslash f(\mathbf{x}_i, \omega_q) \leq f(\mathbf{x}_i, \omega_r)| \tag{1.11}$$

where $\overline{Y}_i$ is the complementary set of Y_i in $\mathcal{Y}$.

Average precision: This metric evaluates the average fraction of labels ranked above a particular label $\omega \in Y_i$ which are actually in Y_i.

$$\mathcal{A}vPrec(f, \mathcal{S}) = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{|Y_i|} \sum_{\omega_q \in Y_i} \frac{|\{\omega_r \in Y_i\} \backslash rank_f(\mathbf{x}_i, \omega_r) \leq rank_f(\mathbf{x}_i, \omega_q)|}{rank_f(\mathbf{x}_i, \omega_q)}. \tag{1.12}$$

Note that $AvPrec(f, \mathcal{S}) = 1$ means that the labels are perfectly ranked. For the other metrics, smaller values correspond to a better label ranking quality.

1.7 Conclusion

This chapter has presented a literature survey on classification, which is traditionally a supervised learning task. We have categorized supervised classification problems into four types depending on the input example, its output target class (or classes) and the mutual relation between the predefined classes.

After that, we have focused our study on multi-label classification. An analysis of the existing multi-label methods has been presented. These methods can be divided in three categories depending on the way they use multi-labeled data. Finally, we have described several real-world applications of multi-label learning.

To evaluate multi-label classifiers, various metrics can be involved. These metrics can be divided into groups: prediction-based and ranking-based metrics. The former is calculated based on the comparison between the predicted labels and the ground truth labels, while the latter is based on the predicted ranking of labels.

Chapter 2

Transferable belief model

2.1 Introduction

Dempster-Shafer theory, also known as the theory of belief functions, was originally introduced by A. Dempster in the 1960's and refined by G. Shafer in the 1970's. In the 1990's, this theory was developed by Smets, who introduced several tools for information fusion and decision making [84]. Nowadays, this theory is widely used to represent uncertain knowledge. However, it suffers from high complexity in applications that have large frames of discernment.

The authors in [25][26] have introduced an approach which makes it possible to define and manipulate belief functions in a very large frame. This approach is based on defining belief functions on a class $\mathcal{C}(\Omega)$ of subsets of 2^{Ω}. It can be used for applying belief functions to multi-label learning problems with uncertainties and ambiguities problems.

In this chapter, we will recall the basic concepts of Dempster-Shafer theory in Section 2.2. We will first describe some elements of this theory namely the frame of discernment as well as mass, belief, commonality and plausibility functions. Secondly, we will discuss different rules for combining items of evidence and making decision. In Section 2.3, we will explain the application of belief functions in $\mathcal{C}(\Omega)$, which is a subset of 2^{Ω}; we will review some basic definitions about lattices and belief functions in lattices. We will also show that the Dempster-Shafer calculus can be extended to the restricted set $\mathcal{C}(\Omega)$, which is close under intersection and has a lattice structure. Direct application of this theory to multi-label classification leads to the Evidential Multi-label

k-Nearest Neighbor (EMLkNN) method, which will be detailed in Section 2.4. Finally, Section 2.5 will conclude the chapter.

2.2 Dempster-Shafer theory

In his book *A Mathematical Theory of Evidence* [80], Shafer introduced the basic concepts of belief functions. In this section, we will present a summary of this theory, followed by the basis operations of combination and making decision.

Frame of discernment

Dempster-Shafer (D-S) theory is a mathematical theory of evidence. It measures the degree of support of a piece of evidence to various propositions by assigning masses between zero and one to subsets of a domain of interest Ω, called the frame of discernment. This theory extends probability theory, in so far as masses are assigned to arbitrary subsets of Ω and not only to singletons. In particular, complete ignorance corresponds to the case where all the mass is assigned to the whole frame of discernment.

Mass function

To express the belief among the different subsets of Ω, a function $m : 2^{\Omega} \rightarrow [0, 1]$ is called a *mass function* verifying:

$$\sum_{A \subseteq \Omega} m(A) = 1. \tag{2.1}$$

The quantity $m(A)$ measures the belief that is committed to A and to no smaller subset. The subsets A of Ω verifying $m(A) > 0$ are called *focal elements* of m. m is called *normalized* if $\emptyset$ is not a focal element, and *dogmatic* if Ω is not a focal element.

Belief function

A mass function induces a belief function defined as follows for all $A \subseteq \Omega$:

$$bel(A) = \sum_{\emptyset \neq B \subseteq A} m(B), \tag{2.2}$$

$bel(A)$ is interpreted as the total degree of support in proposition A, or the degree of belief in this proposition.

Commonality function

Another function in the D-S theory is the *commonality*. The *commonality function* of A, which is the sum of masses allocated to supersets of A, can be interpreted as the total probability mass that can move freely to every point of A. It is equal to:

$$q(A) = \sum_{B \supseteq A} m(B). \tag{2.3}$$

Plausibility function

The *plausibility*, $pl(A)$, quantifies the maximum amount of potential specific support that could be given to A. It is equal to one minus the belief in $\bar{A}$:

$$\begin{aligned} pl(A) &= \sum_{B \cap A \neq \emptyset} m(B) & (2.4)\\ &= 1 - bel(\overline{A}), & (2.5) \end{aligned}$$

where $\bar{A}$ represents the complement of A.

Rules of combination

Let m_1 and m_2 be two mass functions on Ω induced by two different sources of information. The *conjunctive sum* (or unnormalized Dempster's rule of combination) of m_1 and m_2 is a new mass function given by:

$$(m_1 \bigcirc\!\!\!\!\cap\, m_2)(A) = \sum_{B \cap C = A} m_1(B) m_2(C), \tag{2.6}$$

For all $A \subseteq \Omega$, this rule is commutative, associative, and admits the vacuous mass function (defined by $m(\Omega) = 1$) as neutral element. It is conjunctive as the product of $m_1(B)$ and $m_2(C)$ is transferred to the intersection of B and C. The quantity $(m_1 \bigcirc\!\!\!\!\cap\, m_2)(\emptyset)$ is referred to as the degree of conflict.

It is known that the Dempster's rule of combination can also be computed through the commonality function [80]. Given q_1 and q_2, the commonality functions associated to m_1 and m_2, the conjunctive rule in terms of commonality functions is defined by:

$$q_{1 \bigcirc\!\!\!\!\cap\, 2}(A) = q_1(A).q_2(A). \tag{2.7}$$

The normalized Dempster's rule is defined by:

$$(m_1 \oplus m_2)(A) = \begin{cases} \frac{(m_1 \bigcirc\!\!\!\!\cap\, m_2)(A)}{1-K} & \text{if } A \neq \emptyset \\ 0 & \text{if } A = \emptyset, \end{cases} \tag{2.8}$$

where $K = (m_1 \bigcirc\!\!\!\!\cap\, m_2)(\emptyset)$ is the degree of conflict. It is clear that $m_1 \oplus m_2$ is defined as long as $K < 1$.

Let $q_{1\oplus 2}$ denote the commonality function corresponding to $m_1 \oplus m_2$. It can be computed from q_1 and q_2, as follows:

$$(q_1 \oplus q_2)(A) = \begin{cases} \frac{(q_1(A).q_2(A)}{1-K} & \text{if } A \neq \emptyset \\ 1 & \text{if } A = \emptyset. \end{cases} \tag{2.9}$$

Another rule for combination is the disjunctive rule, defined by:

$$(m_1 \bigcirc\!\!\!\!\cup\, m_2)(A) = \sum_{B \cup C = A} m_1(B) m_2(C). \tag{2.10}$$

This rule is also commutative and associative.

Denoting by $bel_1 \bigcirc\!\!\!\!\cup\, bel_2$ the belief function corresponding to $m_1 \bigcirc\!\!\!\!\cup\, m_2$, it can be shown that:

$$(bel_1 \bigcirc\!\!\!\!\cup\, bel_2)(A) = bel_1(A) bel_2(A), \quad \forall A \in \Omega, \tag{2.11}$$

which is the counterpart of (2.9).

In general, the conjunctive rule is used when the pieces of evidence to be combined are given by reliable sources of information while the disjunctive rule is used when at least one of the sources of information is reliable. Both of them assume that the sources of information are independent.

Dubois and Prade [30] have proposed a *hybrid* rule intermediate between the conjunctive and disjunctive sums, in which the product $m_1(B)m_2(C)$ is assigned to $B \cap C$ whenever $B \cap C \neq \emptyset$, and to $B \cup C$ otherwise. This rule is not associative, but it usually provides a good summary of partially conflicting items of evidence.

In [24] a combination rule, called *cautious conjunctive* rule, is introduced to combine belief functions induced by reliable, but possibly overlapping bodies of evidence. This rule is based on the canonical decomposition of belief functions. A disjunctive counterpart to this rule, called the *bold disjunctive* rule can also be defined. Both rules are commutative, associative and idempotent.

Making decision

In order to make decisions, a two-level model is proposed in [84]: the credal level and the pignistic one. At the credal level, items of evidence are represented by mass (or equivalently belief) functions and combined. At the pignistic level, decisions are made by maximizing expected utilities [75]. Once a decision has to be made, a mass function m is transformed into a *pignistic probability distribution* p in order to compute these expectations. The pignistic transformation consists in normalizing m (assuming that $m(\emptyset) < 1$), and then distributing each normalized mass $m(A)/(1 - m(\emptyset))$ equally between the atoms $\omega_q \in A$:

$$p(\omega_q) = \sum_{\{A \subseteq \Omega, \omega_q \in A\}} \frac{m(A)}{(1 - m(\emptyset))|A|}, \quad \forall \omega_q \in \Omega. \tag{2.12}$$

In [19], an alternative method is advocated to transform a belief function into a corresponding probability function model by normalizing the plausibilities of singletons. The plausibility transformation can be used for making decisions with belief functions by selecting the hypothesis with maximum plausibility. The plausibility transformation Pl_p is then defined by:

$$Pl_p(\omega_q) = \frac{pl(\omega_q)}{\sum\{pl(\omega_q)|\omega_q \in \Omega\}}, \quad \forall \omega_q \in \Omega. \tag{2.13}$$

Example 1 Let us assume that a crime has been committed and there are three suspects: Peter, Paul and Mary. Suppose we have two testimonies: Witness 1 says the killer was a man, while witness 2 claims that he has found a blond hair at the crime scene. We know that Mary and Paul are blond.

The frame of discernment for this problem is: $\Omega = \{Peter, Paul, Mary\}$. We know that witness 1 is drunk 20% of the time. This piece of evidence can be represented by a mass function m_1 as follows: $m_1(\{Peter, Paul\}) = 0.8$, $m_1(\{Peter, Paul, Mary\}) = 0.2$. The .8 belief mass given to $\{Peter, Paul\}$ corresponds to the part of belief that witness 1 was not drunk. The rest of belief (1-0.8) is given to the ignorance (Ω): if witness 1 was drunk, we have no information about the killer.

Considering the second item of evidence, there is a probability 0.6 that the room has been cleaned before the crime. The resulting belief function will then be: $m_2(\{Paul, Mary\}) = 0.6$, $m_2(\{Peter, Paul, Mary\}) = 0.4$. Assuming that these two items of evidence are distinct, they should be combined using the conjunctive or disjunctive sum. Table 2.1 represents the items of evidence associated with each witness, with their conjunctive

Table 2.1: Conjunctive and disjunctive combination sums of m_1 and m_2 in Example 1.

Suspect	m_1	m_2	$m_{1\circledcap 2}$	$m_{1\circledcup 2}$
$\emptyset$	0	0	0	0
Paul	0	0	0.48	0
Peter	0	0	0	0
Mary	0	0	0	0
Peter or Paul	0.8	0	0.32	0
Paul or Mary	0	0.6	0.12	0
Mary or Peter	0	0	0	0
Peter, Paul or Mary	0.2	0.4	0.08	1

Table 2.2: Pignistic probability and plausibility transformation associated with the conjunctive combination sum of m_1 and m_2 in Example 1.

Suspect	m_1	m_2	$p_{1\circledcap 2}$	$Plp_{1\cap 2}$
$\emptyset$	0	0	0	0
Paul	0	0	0.727	0.625
Peter	0	0	0.187	0.250
Mary	0	0	0.087	0.125
Peter or Paul	0.8	0	-	-
Paul or Mary	0	0.6	-	-
Mary or Peter	0	0	-	-
Peter, Paul or Mary	0.2	0.4	-	-

and disjunctive sums. The proposition $\emptyset$ means here that none of the three suspects was the killer.

We may remark here, that the disjunctive sum yields to total ignorance by associating a mass function equal to 1 to Ω. For that reason, we should use the conjunctive sum to make a decision and select the element having the maximum pignistic probability or the maximum plausibility. Table 2.2 displays the pignistic probability and the plausibilities of singletons associated with the conjunctive combination. In this case we can remark that the decision given by the two decision rules coincide.

Example 2 Consider now the classification problem and let $\Omega = \{\omega_1, \dots, \omega_Q\}$ be the set of target classes. In the case of multi-class classification, target classes are mutually exclusive and the frame of discernment is usually defined as the set of distinct number of classes is Ω. Mass function is then the mapping from the power set of Ω, $2^\Omega = \{\{\omega_1\}, \ldots, \{\omega_Q\}, \{\omega_1, \omega_2\}, \ldots, \{\omega_{Q-1}, \omega_Q\}, \{\omega_1, \omega_2, \omega_3\}, \ldots, \{\omega_1, \ldots, \omega_Q\}\}$, containing 2^Q elements, to the unit interval. In the case of multi-label learning, target classes are not mutually exclusive because an instance may belong to several classes simultaneously. The frame of discernment will then contain all possible subsets of Ω and will be equal to the power set of Ω, $\Theta = 2^\Omega$. To define mass functions, we have to manipulate subsets of Θ. As there are 2^{2^Q} such subsets, this approach rapidly becomes intractable as the number Q of classes increases. As explained in the introduction, the authors in [25][26] present a formalism for applying the Dempster-Shafer framework to very large frames such as the power set of a finite set Ω with manageable complexity. This formalism will be recalled in the next section.

2.3 Belief functions on set-valued variables

Let Ω be a finite set. Defining belief functions on 2^Ω would lead to double-exponential complexity, and may become intractable except for sets Ω with very small cardinality. However, as shown in [61], the Dempster-Shafer theory can be used with a strict subset $\mathcal{C}(\Omega)$ of 2^Ω that is closed under intersection. The closure system $(\mathcal{C}(\Omega), \subseteq)$ has a lattice structure.

In the following, we will recall the necessary background on lattices following by the application of the theory of belief functions on lattices. The main concepts of Dempster-Shafer will then be extended to the case where we want to describe the uncertainty regarding a set-valued variable V on a finite domain Ω.

2.3.1 Belief functions in general lattices

Lattices

A detailed review of lattice theory can be found in [61]. The following presentation follows [40]. Let L be a finite set and $\leq$ a partial ordering (i.e., a reflexive, antisymmetric and transitive relation) on L. The structure $(L, \leq)$ is called a *poset*. We say that $(L, \leq)$ is a *lattice* if, for every $x, y \in L$, there is a unique greatest lower bound (denoted by $x \wedge y$) and a unique least upper bound (denoted by $x \vee y$). Operations $\wedge$ and $\vee$ are called the *meet* and *join* operations, respectively. For finite lattices, the greatest element $\top$ and the least element $\perp$ always exist. We say that x *covers* y if $x > y$ and there is no z such that $x > z > y$. An element x of L is an *atom* if it covers only one element and this element is $\bot$. It is a *co-atom* if it is covered by a single element and this element is $\top$.

Two lattices L and L' are *isomorphic* if there exists a bijective mapping f from L to L' such that $x \leq y \Leftrightarrow f(x) \leq f(y)$. For any poset $(L, \leq)$, we can define its dual $(L, \geq)$ by inverting the order relation. A lattice is *autodual* if it is isomorphic to its dual.

A lattice is *distributive* if $(x \vee y) \wedge z = (x \wedge z) \vee (y \wedge z)$ holds for all $x, y, z \in L$. For any $x \in L$, we say that x has a complement in L if there exists $x' \in L$ such that $x \wedge x' = \bot$ and $x \vee x' = \top$. L is said to be *complemented* if any element has a complement. A distributive lattice in which each element has a complement is called a Boolean lattice. Every Boolean lattice is isomorphic to $(2^\Omega, \subseteq)$ for some set Ω. For the lattice $(2^\Omega, \subseteq)$, we have $\wedge = \cap$, $\vee = \cup$, $\bot = \emptyset$ and $\top = \Omega$.

A *closure system* $\mathcal{C}$ on a set Θ is a family of subsets of Θ that closed under intersection. As shown in [61], any closure system $(\mathcal{C}, \subseteq)$ is a lattice with the following meet and join operations

$$C_1 \wedge C_2 = C_1 \cap C_2 \tag{2.14}$$

$$C_1 \vee C_2 = \bigcap \{C \in \mathcal{C} | C_1 \cup C_2 \subseteq C\}. \tag{2.15}$$

Belief functions on lattices

As shown in [40], the theory of belief function can be extended from the Boolean lattice $(2^\Omega, \subseteq)$ to any lattice, not necessarily Boolean. Most results of Dempster-Shafer theory can be transposed in the general lattice setting $(L, \leq)$. For instance, Dempster's rule can be extended by replacing $\cap$ by $\wedge$ in (2.8), and relation (2.9) between commonality functions is preserved. Similarly, we can extend the disjunctive rule (2.10) by substituting $\vee$ for $\cup$ in (2.10), and relation (2.11) still holds. The extension of other notions from classical Dempster-Shafer theory may require additional assumptions on $(L, \leq)$. For instance, the definition of the plausibility function *pl* as the dual of *bel* using (2.5) can only be extended to autodual lattices. Also, probability measures cannot be defined on arbitrary lattices. Consequently, the pignistic probability (2.12) can only be extended in restricted settings.

2.3.2 Belief functions on the lattice $(\mathcal{C}(\Omega), \subseteq)$

The Lattice $(\mathcal{C}(\Omega), \subseteq)$

Let V be a variable taking zero, one or several values in a finite set Ω. Such a variable is said set-valued [26].

Let $A_0 \subseteq \Omega$ be the unknown true value of V. To express partial knowledge about a set-valued variable V in the belief function framework, we should define a mass function m^Θ on Θ. As explained in the beginning of this section, this approach is inapplicable except for sets Ω with very small cardinality.

As an alternative, the authors in [26] proposed to define mass functions and associated functions on a subset of 2^Θ that forms a lattice when equipped with the inclusion relation. The intuitive idea underlying this approach is the fact that, when expressing knowledge about a set-valued variable V, it is often convenient to specify sets of values that are *certainly* taken by V, and sets of values that are *certainly not* taken by V. This can be illustrated by the following example.

Example 3 Let V denote the emotions evoked upon hearing a song, defined on the large set Ω of existing emotions. If an expert may decide that this song certainly evokes happiness and certainly does not evoke sadness, but may be undecided regarding the other emotions (such as quietness, anger, surprise, etc.), then all subsets of Ω containing $A = \{\textit{happiness}\}$ and not intersecting $B = \{\textit{sadness}\}$ are possible values of V.

More generally, let $\mathcal{Q}(\Omega) = \{(A, B) \in 2^\Omega \times 2^\Omega | A \cap B = \emptyset_\Omega\}$ be the set of ordered pairs of disjoint subsets of Ω, where $\emptyset_\Omega$ denotes the empty set of Ω. For any $(A, B) \in \mathcal{Q}(\Omega)$, let $\varphi(A, B)$ denote the following subset of $\Theta = 2^\Omega$:

$$\varphi(A, B) = \{C \subseteq \Omega | C \supseteq A, C \cap B = \emptyset_\Omega\}. \tag{2.16}$$

$\varphi(A, B)$ is thus the subset of Θ composed of all subsets of Ω including A and non intersecting B. Equivalently, it is the set of all subsets of Ω that include A and are included in $\overline{B}$:

$$\varphi(A, B) = \{C \subseteq \Omega | A \subseteq C \subseteq \overline{B}\}. \tag{2.17}$$

It is thus the interval $[A, \overline{B}]$ in the lattice $(2^\Omega, \subseteq)$.

Let $\mathcal{C}(\Omega)$ denote the set of all subsets of Θ of the form $\varphi(A, B)$, completed by the empty set of Θ, noted $\emptyset_\Theta$:

$$\mathcal{C}(\Omega) = \{\varphi(A, B) | A \subseteq \Omega, B \subseteq \Omega, A \cap B = \emptyset_\Omega\} \cup \{\emptyset_\Theta\}.$$

$\mathcal{C}(\Omega)$ is thus a subset of 2^Θ. For a reason that will become evident later, we will also use $\varphi(\Omega,\Omega)$ as an alternative notation for $\emptyset_\Theta$. Function φ is thus a bijective mapping from $\mathcal{Q}^*(\Omega) = \mathcal{Q}(\Omega) \cup \{(\Omega,\Omega)\}$ to $\mathcal{C}(\Omega)$. The following proposition states that $\mathcal{C}(\Omega)$ is a closure system and, consequently, has a lattice structure.

Proposition 1 $\mathcal{C}(\Omega)$ is a closure system of Θ, and

$$\varphi(A,B) \cap \varphi(A',B') = \begin{cases} \varphi(A \cup A', B \cup B') & \text{if } (A \cup A') \cap (B \cup B') = \emptyset_\Omega \\ \emptyset_\Theta & \text{otherwise,} \end{cases}$$

for all (A,B) and (A',B') in $\mathcal{Q}^*(\Omega)$.

The meet operation is the intersection, and the join operation $\sqcup$ is defined by:

$$\varphi(A,B) \sqcup \varphi(A',B') = \varphi(A \cap A', B \cap B').$$

As noticed in [41], any ordered pair (A,B) of disjoint subsets of $\Omega = \{\omega_1,\ldots,\omega_Q\}$ can be represented by a vector $(y_1,\ldots,y_Q) \in \{-1,0,1\}^Q$, with

$$y_i = \begin{cases} 1 & \text{if } \omega_i \in A, \\ -1 & \text{if } \omega_i \in B, \\ 0 & \text{otherwise.} \end{cases}$$

This encoding makes it clear that the cardinality of $\mathcal{C}(\Omega)$ is equal to $3^{|\Omega|}+1$ which is much less than the $2^{2^{|\Omega|}}$ elements of 2^Θ.

Belief functions in $(\mathcal{C}(\Omega),\subseteq)$

The general theory of belief functions can then be applied directly to the special lattice $(\mathcal{C}(\Omega),\subseteq)$.

Let $m : \mathcal{C}(\Omega) \to [0,1]$ be a mass function on $\mathcal{C}(\Omega)$. The notation $m(\varphi(A,B))$ will be simplified to $m(A,B)$. For this reason, m will be called a *two-place mass function.*

Belief and commonality functions can be computed from m using the following formula:

$$\begin{aligned} bel(A,B) &= \sum_{\varphi(C,D)\subseteq\varphi(A,B)} m(C,D) - m(\Omega,\Omega) && (2.18)\\ &= \sum_{C\supseteq A, D\supseteq B} m(C,D) - m(\Omega,\Omega), && (2.19)\\ q(A,B) &= \sum_{\varphi(C,D)\supseteq\varphi(A,B)} m(C,D) && (2.20)\\ &= \sum_{C\subseteq A, D\subseteq B} m(C,D). && (2.21)\end{aligned}$$

The conjunctive sum operation in $\mathcal{C}(\Omega)$ is defined as follows:

$$\begin{aligned} (m_1 \circledcap m_2)(A,B) &= \sum_{\varphi(C,D)\cap\varphi(E,F)=\varphi(A,B)} m_1(C,D)m_2(E,F) && (2.22)\\ &= \begin{cases} \sum_{C\cup E=A, D\cup F=B} m_1(C,D)m_2(E,F) & \text{if } A\cap B=\emptyset_\Omega,\\ \sum_{(C\cup E)\cap(D\cup F)\neq\emptyset_\Omega} m_1(C,D)m_2(E,F) & \text{if } A=B=\Omega. \end{cases} && (2.23)\end{aligned}$$

It can be computed using the commonality functions as:

$$q_{1\circledcap 2}(A,B) = q_1(A,B)\cdot q_2(A,B), \quad \forall (A,B)\in\mathcal{Q}^*(\Omega). \qquad (2.24)$$

The disjunctive sum can be defined as follows:

$$\begin{aligned} (m_1 \circledcup m_2)(A,B) &= \sum_{\varphi(C,D)\sqcup\varphi(E,F)=\varphi(A,B)} m_1(C,D)m_2(E,F) && (2.25)\\ &= \sum_{C\cap E=A, D\cap F=B} m_1(C,D)m_2(E,F). && (2.26)\end{aligned}$$

Another equation allows us to calculate m from q:

$$m(A,B) = \sum_{C\subseteq A, D\subseteq B} (-1)^{|A\setminus C|+|B\setminus D|} q(C,D). \qquad (2.27)$$

Making decision

When working with belief functions in a Boolean Lattice $(2^\Omega, \subseteq)$, a usual decision rule is to select the singleton $\{\omega\}$ of Ω with the largest plausibility or, equivalently, with the largest commonality. In the lattice $(\mathcal{C}(\Omega), \subseteq)$, the plausibility function is not defined, but the commonality function exists and its maximum can be easily computed by solving an integer programming problem with non-linear constraints. A possible rule for decision making is thus to select the element of Ω with the largest commonality [25].

Another rule for making decision is to compute, for each item $\{\omega\}$ in Ω, two degrees of belief: $bel(\{\omega\}, \emptyset)$ and $bel(\emptyset, \{\omega\})$ (like the Dempster-Shafer theory when making decision at the credal level). If $(bel(\{\omega\}, \emptyset) \geq bel(\emptyset, \{\omega\}))$, then the final decision must contain $\{\omega\}$ [26].

Example 4 Let $\Omega = \{\omega_1, \omega_2, \omega_3\}$ be the set of classes for a multi-label classification problem. Let $\mathbf{x}$ be an unseen instance to classify. Suppose that we receive the following items of evidence through two experts' opinions:

1. Expert 1 tells us that $\mathbf{x}$ certainly belongs to class $\{\omega_1\}$ and certainly not belong to class $\{\omega_3\}$, with confidence 0.8. This is represented by the following mass function:
$$m_1(\{\omega_1\}, \{\omega_3\}) = 0.80, \quad m_1(\emptyset, \emptyset) = 0.20.$$
2. Expert 2 tells us that $\mathbf{x}$ belongs either to $\{\omega_1\}$ or $\{\omega_2\}$. He is sure at 70% that $\mathbf{x}$ should be assigned to class ω_1, and with a certainty equal to 15% that $\mathbf{x}$ should be assigned to class ω_2. This is represented by:
$$m_2(\{\omega_1\}, \emptyset\}) = 0.70, \quad m_2(\{\omega_2\}, \emptyset) = 0.15, \quad m_2(\emptyset, \emptyset) = 0.15.$$

Assuming these two items of evidence to be distinct, they should be combined using the conjunctive sum operation $\textcircled{\cap}$. This may be achieved in two ways:

1. We may compute the intersection between each focal element of m_1 and each focal element of m_2 and apply formula (2.22). The computations may be presented as in Table 2.3.
2. Alternatively, we may compute the commonality functions q_1 and q_2 using (2.20), multiply them, and convert the result into a mass function using (2.27). The intermediate and final results are shown in Table 2.4.

We may check that both approaches yield to the same result. Let $m_{12} = m_1 \textcircled{\cap} m_2$. We get:

$$\begin{aligned} m_{12}(\{\omega_1\}, \{\omega_3\}) &= 0.68, \\ m_{12}(\{\omega_1\}, \emptyset) &= 0.14, \\ m_{12}(\{\omega_1, \omega_2\}, \{\omega_3\}) &= 0.12, \\ m_{12}(\{\omega_2\}, \emptyset) &= 0.03, \\ m_{12}(\emptyset, \emptyset) &= 0.14. \end{aligned}$$

Table 2.3: Computation of the conjunctive sum of m_1 and m_2 in Example 4. The columns and the lines correspond to the focal elements of m_1, and m_2, respectively. Each cell contains the intersection of a focal element of m_1 and a focal element of m_2. The mass of each focal element is indicated below it.

	$(\{\omega_1\},\{\omega_3\})$ 0.8	$(\emptyset,\emptyset)$ 0.2
$(\{\omega_1\},\emptyset)$ 0.7	$(\{\omega_1\},\{\omega_3\})$ 0.7×0.8	$(\{\omega_1\},\emptyset)$ 0.7×0.2
$(\{\omega_2\},\emptyset)$ 0.15	$(\{\omega_1,\omega_2\},\{\omega_3\})$ 0.15×0.8	$(\{\omega_2\},\emptyset)$ 0.15×0.2
$(\emptyset,\emptyset)$ 0.15	$(\{\omega_1\},\{\omega_3\})$ 0.15×0.8	$(\emptyset,\emptyset)$ 0.15×0.2

Based on this evidence and in order to make a decision, we select from Table 2.4 the singleton with the maximum of communality, which is $\{\omega_1,\omega_2\}$. Using the second rule of decision, we should compare two quantities($bel(\{\omega\},\emptyset)$ and $bel(\emptyset,\{\omega\})$) according to Equation (4.5). From Table 2.5, we can say that $\mathbf{x}$ can be labelled with $\{\omega_1,\omega_2\}$ but not with $\{\omega_3\}$.

2.4 Application to multi-label classification

Consider the multi-label classification problem, in which objects may belong simultaneously to several classes. In order to use the formalism presented in the previous section, as explained in [26], the training set is then presented by $\mathcal{D} = \{(\mathbf{x}_1, A_1, B_1), \ldots, (\mathbf{x}_n, A_n, B_n)\}$, where $A_i \subseteq \mathcal{Y}$ denotes a set of classes that surely apply to instance $\mathbf{x}_i$, and $B_i \subseteq \Omega$ a set of classes that surely do not apply to the same instance. If $Y_i \subseteq \mathcal{Y}$ denotes the true label set of $\mathbf{x}_i$, we thus only know that $Y_i \in \varphi(A_i, B_i)$. The Evidential Multi-label k-Nearest Neighbor (EMLkNN) method builds a multi-label classifier $\mathcal{H}$ as will be explained in the following.

Given a new instance $\mathbf{x}$, let $\mathcal{N}_{\mathbf{x}}^k$ be the set of its k nearest neighbors in $\mathcal{D}$, according to some distance d. This item of evidence given by $(\mathbf{x}_i, A_i, B_i)$, where $\mathbf{x}_i$ is an element of that set, is represented by the following mass function:

Table 2.4: Computation of $m_1 \textcircled{\cap} m_2$ and $m_1 \oplus m_2$ in Example 4.

A	B	m_1	q_1	m_2	q_2	$q_{1\textcircled{\cap}2}$	$m_1 \textcircled{\cap} m_2$	$m_1 \oplus m_2$
$\emptyset$	$\emptyset$	0.2	1	0.15	0.15	0.15	0.03	0.03
$\emptyset$	$\{\omega_1\}$	0	0.2	0	0.15	0.03	0	0
$\emptyset$	$\{\omega_2\}$	0	0.2	0	0.15	0.03	0	0
$\emptyset$	$\{\omega_3\}$	0	0.2	0	0.15	0.03	0	0
$\emptyset$	$(\{\omega_1, \omega_2\})$	0	0.2	0	0.15	0.03	0	0
$\emptyset$	$(\{\omega_1, \omega_3\})$	0	0.2	0	0.15	0.03	0	0
$\emptyset$	$(\{\omega_2, \omega_3\})$	0	0.2	0	0.15	0.03	0	0
$\emptyset$	$(\{\omega_1, \omega_2, \omega_3\})$	0	0.2	0	0.15	0.03	0	0
$\{\omega_1\}$	$\emptyset$	0	0.2	0.7	0.85	0.17	0.14	0.14
$\{\omega_1\}$	$\{\omega_2\}$	0	0.2	0	0.85	0.17	0	0
$\{\omega_1\}$	$\{\omega_3\}$	0.8	1	0	0.85	0.85	0.68	0.68
$\{\omega_1\}$	$(\{\omega_2, \omega_3\})$	0	1	0	0.85	0.85	0	0
$\{\omega_2\}$	$\emptyset$	0	0.2	0	0.30	0.06	0.03	0.03
$\{\omega_2\}$	$\{\omega_1\}$	0	0.2	0.15	0.30	0.06	0	0
$\{\omega_2\}$	$\{\omega_3\}$	0	0.2	0	0.30	0.06	0.015	0.015
$\{\omega_2\}$	$(\{\omega_1, \omega_3\})$	0	0.2	0	0.30	0.06	0.015	0.015
$\{\omega_3\}$	$\emptyset$	0	0.2	0	0.15	0.03	0	0
$\{\omega_3\}$	$\{\omega_1\}$	0	0.2	0	0.15	0.03	0.07	0.07
$\{\omega_3\}$	$\{\omega_2\}$	0	0.2	0	0.15	0.03	0	0
$\{\omega_3\}$	$(\{\omega_1, \omega_2\})$	0	0.2	0	0.15	0.03	0	0
$(\{\omega_1, \omega_2\})$	$\emptyset$	0	0.2	0	1	0.2	0	0
$(\{\omega_1, \omega_2\})$	$\{\omega_3\}$	0	1	0	1	1	0.12	0.12
$(\{\omega_1, \omega_3\})$	$\emptyset$	0	0.85	0	0.85	0.17	0	0
$(\{\omega_1, \omega_3\})$	$\{\omega_2\}$	0	0.85	0	0.85	0.17	0	0
$(\{\omega_2, \omega_3\})$	$\emptyset$	0	0.30	0	0.30	0.06	0	0
$(\{\omega_2, \omega_3\})$	$\{\omega_1\}$	0	0.30	0	0.30	0.06	0	0
$(\{\omega_1, \omega_2, \omega_3\})$	$\emptyset$	0	1	0	1	0.2	0	0
$(\{\omega_1, \omega_2, \omega_3\})$	$(\{\omega_1, \omega_2, \omega_3\})$	0	1	0	1	1	0	0

Table 2.5: Computation of $bel(\{\omega\}, \emptyset)$ and $bel(\emptyset, \{\omega\})$ in Example 4.

ω_i	$bel(\{\omega_i\}, \emptyset)$	$bel(\emptyset, \{\omega_i\})$
ω_1	0.94	0
ω_2	0.15	0
ω_3	0	0.80

$$m_i(A_i, B_i) = \alpha \exp\left(-\gamma d(\mathbf{x}, \mathbf{x}_i)\right), \tag{2.28}$$

$$m_i(\emptyset, \emptyset) = 1 - \alpha \exp\left(-\gamma d(\mathbf{x}, \mathbf{x}_i)\right), \tag{2.29}$$

where α and γ are two parameters, such that $0 < \alpha < 1$ and $\gamma > 0$.

The k mass functions resulting from the neighborhooding of $\mathbf{x}$ may then combined by the conjunctive sum:

$$m = \textcircled{\scriptsize\cap}_{i:x_i \in \mathcal{N}_{\mathbf{x}}^k} m_i. \tag{2.30}$$

For making decision, we can find the set of labels with the greatest communality, or we can compare the two degrees of belief ($bel(\{\omega\}, \emptyset)$ and $bel(\emptyset, \{\omega\})$) for each label $\{\omega\}$. Thus, the multi-label classifier $\mathcal{H}$ is defined by:

$$\mathcal{H}(\mathbf{x}) = \{\omega \in \mathcal{Y} \mid bel(\{\omega\}, \emptyset) \geq bel(\emptyset, \{\omega\})\}. \tag{2.31}$$

2.5 Conclusion

In this chapter, we have presented a review on evidence combination and reasoning under uncertainty using belief functions. The theory of belief functions is a general framework for reasoning with uncertainty; however, its complexity has to be controlled when it is applied to very large frames of discernment. We introduced the formalism for quantifying uncertainty on a set-valued variable V defined on a domain Ω in the belief function framework. When the frame of discernment Ω forms itself a lattice for some partial ordering, the set of events may be defined as a family $\mathcal{C}(\Omega)$ of subsets of 2^{Ω}. Using this method, it is possible to define and manipulate belief functions in very large frames such as the power set of a finite set. In the following chapter, we will use this approach to develop a new multi-label classification method.

Chapter 3

Extension of the *RAKEL* method to set-valued variables

3.1 Introduction

Chapter 1 reviewed the different approaches to multi-label learning using classical classification and ranking. We have shown that there exist three principal categories: problem transformation approaches, problem adaptation algorithms and ensemble methods. In this chapter, we are interested in ensemble methods and especially the *RAKEL* algorithm.

This chapter reviews the *RAKEL* method, which seeks to classify multi-labelled instances while taking the relation between labels into account. However, this method incurs some loss of information since it is based on learning with small subsets of labels. Consequently, the relationships between labels are then not well represented. Moreover, there are three parameters to be specified a priori. To alleviate these problems, we extend this method to the belief functions framework recalled in Chapter 2. The resulting novel algorithm, called *Evidential RAKEL*, addresses some of the above problems while keeping the property of handling correlation between labels.

Section 3.2 reviews the *RAKEL* method and details the corresponding algorithm. Section 3.3 discusses the correlation between labels and expresses this property on some multi-labelled datasets. Section 3.4 introduces the proposed evidential multi-label classification algorithm based on the *RAKEL* approach. Experiments on both synthetic and real-world datasets will be illustrated in Section 3.5. Finally, Section 3.6

concludes this chapter.

3.2 Random K-labEL sets: RAKEL

Let $\mathbb{X}$ denote an instance space and let $\mathcal{Y} = \{\omega_1, \omega_2, \ldots, \omega_Q\}$, be the finite set of labels in a multi-label training task. A training set of n instances is denoted by $\mathcal{D} = \{(\mathbf{x}_i, Y_i), i = 1, \ldots, n\}$, where $\mathbf{x}_i \in \mathbb{X}$ is a feature vector describing instance i, and $Y_i \subseteq \mathcal{Y}$ is the set of labels for that instance. The goal is to output a multi-label classifier $\mathcal{H}$ that predicts a set of labels for each unseen instance.

RAndom k-labEL sets (*RAKEL*) is an ensemble method for multi-label classification. *RAKEL* handles multi-labelled data by generating LP-based multi-label classifiers for different small-size subsets of labels [95][96]. To reduce the computational complexity of the effective LP method for large numbers of labels and training instances, *RAKEL* involves "labels splitting" and "LP classification" and build a set of classifiers from different subsets of the finite set of labels. In order to label unseen instances, the method combines the predictions of these multiple classifiers using a voting process to output the value of each label separately.

Label splitting. Given a size of label sets k, *RAKEL* constructs m random subsets of labels, Z_j, $\{j = 1, \ldots, m\}$, from $\mathcal{Y}$. The different label sets may be overlapping and the overlap is certain when $mk > Q$. For each label set Z_j, the associated training set, denoted as $\mathcal{D}_j$, is obtained from the original training set $\mathcal{D}$ by replacing the label sets of training instances, Y_i, by their intersections with Z_j: $\mathcal{D}_j = \{(\mathbf{x}_i, Y_i \cap Z_j),\ i = 1, \ldots, n\}$. Note that this may lead to examples annotated by the empty set. These examples are not excluded from $\mathcal{D}_j$ but included in another class by considering the empty set as a new class.

LP classification. Each training set $\mathcal{D}_j$ is learnt by a single-label classifier h_j having as class values all the subsets of Z_j that are found in $\mathcal{D}_j$. Given an new instance $\mathbf{x}$, each single-label classifier h_j provides binary predictions ($+1$ or -1) on each label in Z_i. The rest of labels ($\mathcal{Y} \setminus Z_j$) are not learnt by h_j, and their predictions by h_j are denoted by 0.

Fusion of classifier decisions. To predict the set of labels for $\mathbf{x}$, predictions of single-label classifiers are gathered and their mean is calculated separately for each label $\omega_q \in \mathcal{Y}$. An adapted threshold $\mathbf{t}$, usually equal to 0.5, is used in order to give the final decision. This intuitive threshold correspond to the majority voting rule for the fusion of classifier decisions. Thus, the multi-label classifier $\mathcal{H}$ for the *RAKEL* method is determined as follows:

$$\mathcal{H}(\mathbf{x}) = \{\omega_q \in \mathcal{Y} | [(\sum_{j=1}^{m} (\mathrm{h}_j(\mathbf{x}, \omega_q) > 0)) / (\sum_{j=1}^{m} \mid \mathrm{h}_j(\mathbf{x}, \omega_q) \mid] \geq \mathbf{t}\}.$$

This method can predict a label set that was not present in the initial training set because the final output of the multi-label classifier is assembled from different predictions of all single-label classifiers. The number of single-label classifiers m and the number of labels in each model k are tunable parameters that need to be specified. For $k = 1$ and $m = Q$, *RAKEL* trains Q binary classifiers and we get the BR approach, while for $k = Q$ and $m = 1$, we get the single-label classifier of the LP approach.

To improve the performance of the *RAKEL* method, one should increase the expected number of outputs per label, which means that mk should take a large value. Since the complexity of *RAKEL* grows exponentially with the size of label sets k, but only linearly with the number of classifiers m, it is more usual to set k to a small value (e.g. $k = 3$), while giving to m a range of values going from Q to $2 * Q$.

Concerning the threshold parameter, it is usual to consider a range of values from 0 to 1 with a 0.1 step. Experimental results in [96] demonstrate that low $\mathbf{t}$ values lead to better results for some metrics (e.g. *F1*-measure which promotes the intersection between the initial and predicted sets of labels, when $\mathbf{t}$ has a low value, the number of predicted labels increases and *F1* tends to have high value), while large $\mathbf{t}$ values yield better results for other metrics (e.g. Hamming loss). Authors in [95] fix $\mathbf{t}$ to 0.5 on intuitive grounds.

The *RAKEL* algorithm

The *RAKEL* algorithm is summarized in Figure 3.1. It involves three main modules. The first one, "Label splitting", divides the set of labels into m overlapping, equally-sized subsets. The second module, "multi-label learning", uses a LP single-label classifier for each subset. The last module implements the fusion of classifier decisions for the prediction of label sets for a new instance.

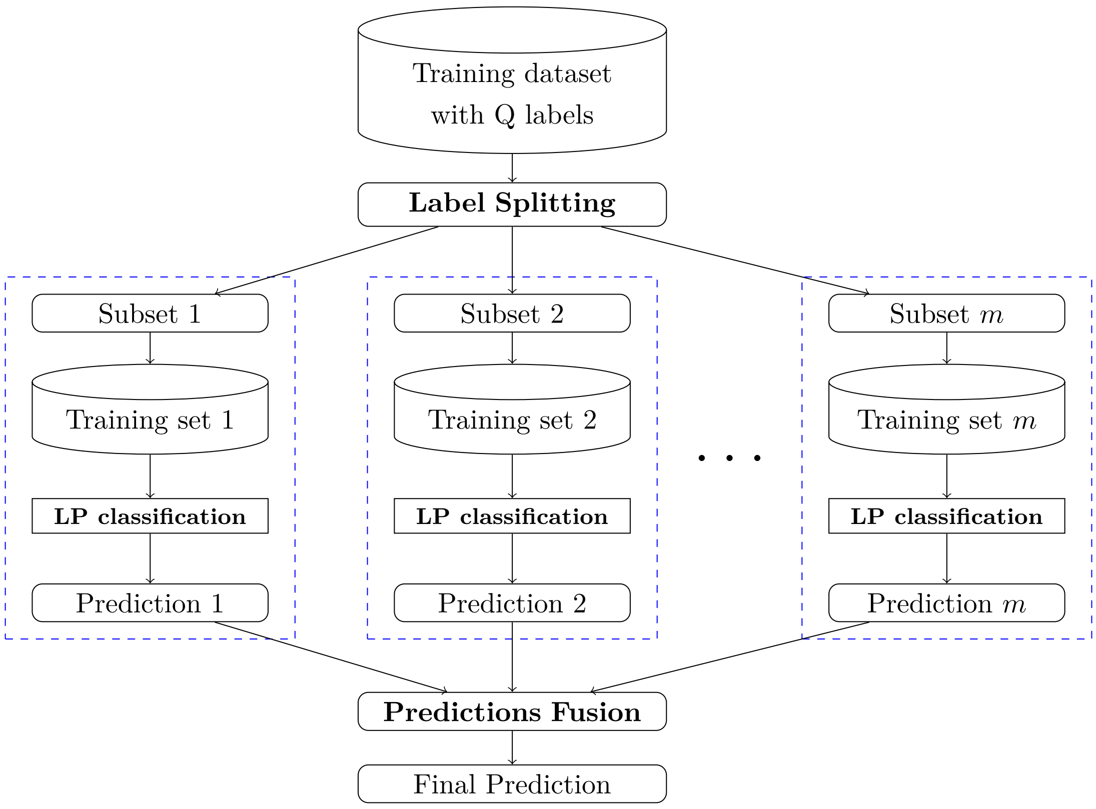


Figure 3.1: The *RAKEL* algorithm

3.3 Correlation between labels

Learning the relationships between labels is a challenging problem in multi-label classification where classes are overlapped and correlated, in the sense that, the annotation of an instance by some class may provide information about the membership of that instance to other classes. In the absence of label relationships, the multi-label task is uninteresting and the given data can be handled by a single-label method without any loss of information.

Various approaches have been proposed in the literature to model the correlation between labels [70][111]. Relations between labels can be exploited via different orders: binary or high order. In the former, label correlations can be expressed as relations between each pair of labels, while the latter captures relations between a label and all remaining labels. High order relations are more complex to represent than binary correlations, which can be measured by the conditional probability of one label given another.

Binary correlations can be visualized in different ways [70]. Given a multi-labelled data $\mathcal{D}$ with Q possible labels, one way to measure the co-occurrence of two labels ω_q and ω_r is through conditional probabilities. The conditional probability of label ω_q being relevant given that the label ω_r is relevant, is $\mathrm{P}(\omega_q|\omega_r)$. $\mathrm{P}(\omega_q)$ is the prior probability. When $\mathrm{P}(\omega_q|\omega_r)$ and $\mathrm{P}(\omega_q)$ are different, one can say that there is some relationship between ω_q and ω_r. Figures 3.2 and 3.3 show the graph of co-occurrences label probabilities for the Emotions ($Q = 6$) and Scene ($Q = 6$) datasets used in our experiments, which will be detailed in Section 4.4.2. In this figure, node thickness indicates prior probability $\mathrm{P}(\omega_q)$, while edge thickness indicates the co-occurrence probability: $\mathrm{P}(\omega_q|\omega_r) \times \mathrm{P}(\omega_r)$, calculated from the training dataset. Figures 3.4 and 3.5 display the contingency matrix M for the same datasets, in which we explore the conditional and prior probabilities for each pair of classes: $\mathrm{M}_{qr} = \mathrm{P}(\omega_q|\omega_r)$ and $\mathrm{M}_{qq} = \mathrm{P}(\omega_q)$. We can see clearly that, for the scene dataset, some labels have more than two relationships, while others like "sunset", have no relationships with any other labels.

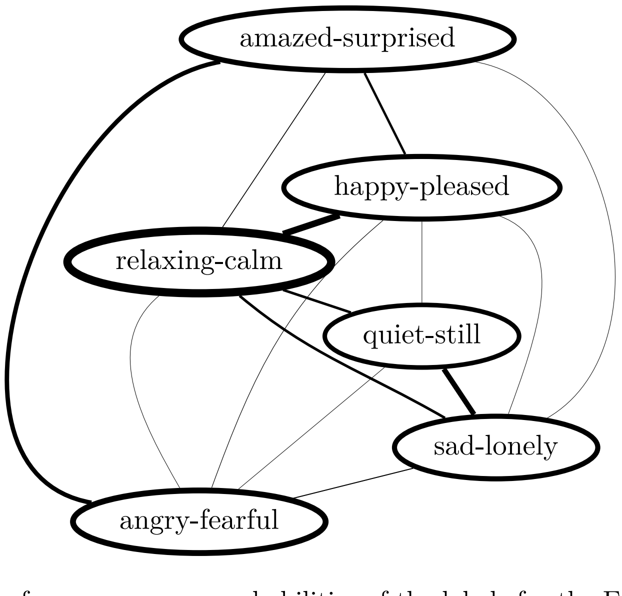


Figure 3.2: Graph of co-occurrences probabilities of the labels for the Emotions dataset

The oldest approach in multi-label learning that considers relation between labels is the LP approach. As explained in Section 1.3, LP is a simple effective multi-label learning method, which considers each unique set of labels in the multi-label learning as a different class of a single-label classification task. LP has the advantage of taking

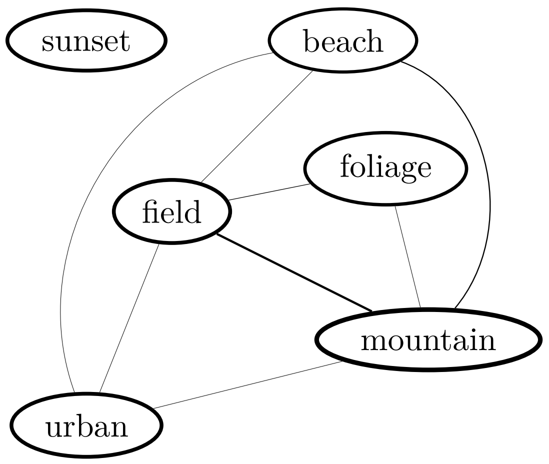


Figure 3.3: Graph of co-occurrences probabilities of the labels for the Scene dataset

correlations among labels into consideration. However, LP may lead to imbalanced datasets with a large number of classes and few examples per class.

RAKEL deals with the aforementioned problems by breaking the large set of labels into different small-sized subsets and employing LP in each of these subsets. To aggregate the sub-classifiers, *RAKEL* uses the voting process to predict the output on each label separately for a given instance. The *RAKEL* method has several advantages: it is simple, effective, and it takes label correlation into account. Additionally, it can be used with any state-of-the-art single-label classification. However, when the initial number of labels Q is large, choosing small value of k has the consequence that correlation between labels is not well taken into account. Hence, we need to increase the value of k. However, the complexity of *RAKEL* is exponential with k. Thus, by aggregating the different sub-classifier results, we may lose some information about the correlation between the initial subsets of labels.

Given a test instance $\mathbf{x}$ to classify with the *RAKEL* method, the output of each sub-classifier is regarded as a set-valued variable by taking zero, one or several values in $\mathcal{Y}$. Table 3.1 represents an example annotated with the *RAKEL* method on a problem where the number of labels is $Q = 6$, using the following parameters: $k = 3$, $m = 5$, and $\mathbf{t} = 0.5$. Each of the 5 classifiers expresses partial knowledge about $\mathbf{x}$, for example, with respect to h_1, ω_1 and ω_3 are *certainly* taken by $\mathbf{x}$, while ω_2 are *certainly not* taken by $\mathbf{x}$, and we have no information concerning the rest of labels (ω_4, ω_5 and ω_6). Moreover, in the final phase, the decision on each label is made independently of the subset of

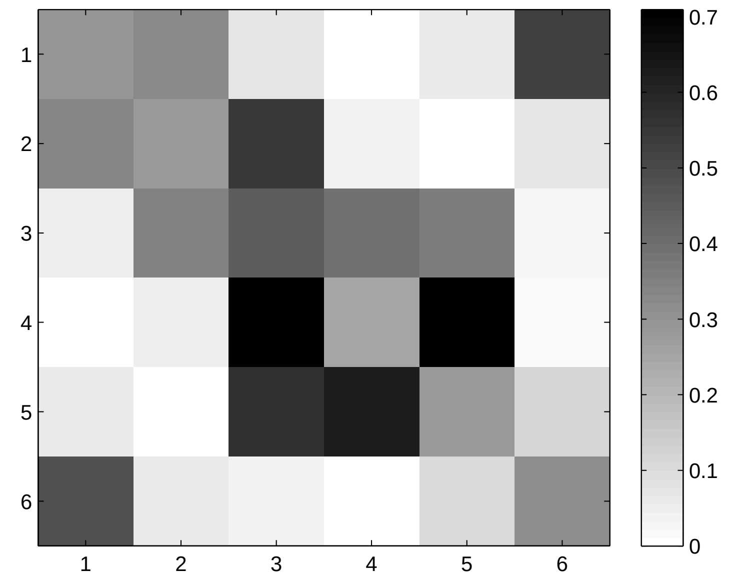


Figure 3.4: Contingency matrix M of prior probabilities: $\mathrm{M}_{qq} = \mathrm{P}(\omega_q)$ (on the diagonal) and conditional probabilities: $\mathrm{M}_{qr} = \mathrm{P}(\omega_q|\omega_r)$ (q is a row and r is a column) for the Emotions dataset

Table 3.1: An example of classification with the *RAKEL* method on a multi-label problem where $Q = 6$, and with $k = 3$, $m = 5$, and $\mathbf{t} = 0.5$.

classifier	3-label sets	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6
h_1	$\{\omega_1, \omega_2, \omega_3\}$	1	-1	1	0	0	0
h_2	$\{\omega_2, \omega_3, \omega_5\}$	0	1	1	0	-1	0
h_3	$\{\omega_3, \omega_5, \omega_6\}$	0	0	-1	0	-1	1
h_4	$\{\omega_1, \omega_4, \omega_6\}$	1	0	0	-1	0	-1
h_5	$\{\omega_4, \omega_5, \omega_6\}$	0	0	0	-1	1	1
	average votes	2/2	1/2	2/3	0/2	1/3	2/3
	final prediction	1	1	1	-1	-1	1

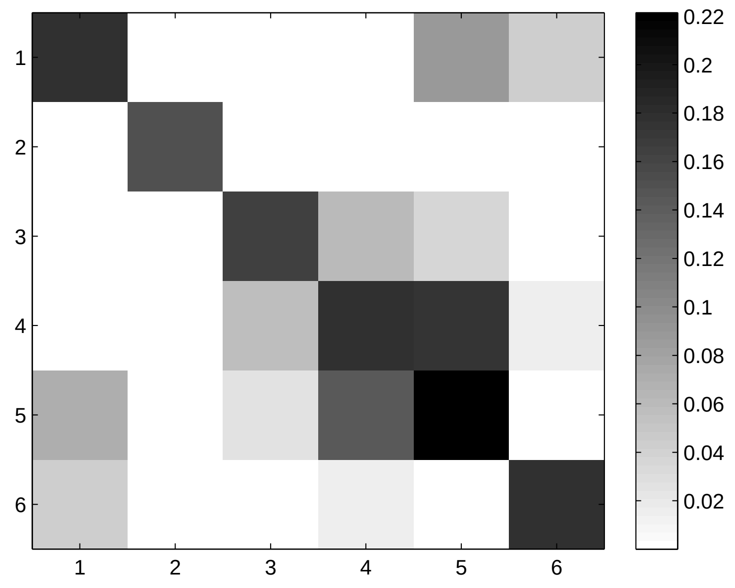


Figure 3.5: Contingency matrix M of prior probabilities: $\mathrm{M}_{qq} = \mathrm{P}(\omega_q)$ (on the diagonal) and conditional probabilities: $\mathrm{M}_{qr} = \mathrm{P}(\omega_q|\omega_r)$ (q is a row and r is a column) for the Scene dataset

labels leading to such result. A more adequate approach for multi-label learning might be to consider results of all classifiers and their corresponding subsets of labels. In this case, the final decision would result not only on the predictions of the base classifiers, but also on their output probabilities.

In this chapter, our purpose is to reduce the loss of information inherent in the *RAKEL* method (as each base classifier only considers a subset of labels), while accounting for label correlation in a more efficient way. Our approach will be based on the theory of evidence. We propose to retain the basic principle of the *RAKEL* approach but to combine the different classifiers in the belief function framework. The proposed approach is called *Evidential RAKEL*. Three major issues are related to our method. The first issue concerns the representation of information from all classifiers regarding the classification of unseen instance $\mathbf{x}$. Each base classifier output is represented by a mass function on the corresponding subset of labels. The second issue refers to the class membership resulting from each classifier. In the framework of set-valued variables, the knowledge about the labelling of an instance can be represented by sets of values certainly taken by $\mathbf{x}$ and sets of values certainly not taken by $\mathbf{x}$ [26]. The third issue concerns the fusion of classifier outputs in order to achieve more reliable information and better decisions.

3.4 Evidential RAKEL

3.4.1 Principle of the Method

Evidential RAKEL is based on the *RAKEL* approach. For each of the subsets $Z_j,\ j \in \{1, \ldots, m\}$, the training phase of the algorithm consists of replacing the label sets of training instances Y_i by their intersections with $Z_j : \mathcal{D}_j = \{(\mathbf{x}_i, Y_i \cap Z_j), i = 1, \ldots, n\}$. Inside $\mathcal{D}_j$, and if the size of $\mathcal{D}_j$ is sufficiently high, we may have until 2^k combinations of labels resulting from applying the LP approach where each combination can be considered as a new class. Thus, *Evidential RAKEL* trains m multi-label classifiers $\mathrm{h}_1, \mathrm{h}_2, \ldots, \mathrm{h}_m$. With this method, the classification task is usually done with the same classification algorithm for the m classifiers.

Given an unseen instance $\mathbf{x}$, each classifier h_j produce probabilities on the subsets of Z_j in order to make a decision about the labelling of $\mathbf{x}$ by h_j. More explicitly, let l be the number of all subsets of Z_j ($l = 2^k$), the output of h_j will be described by a score vector $P_j = (P_{j,1}, P_{j,2}, \ldots, P_{j,l})$, given on a vector gathering all subsets of Z_j, which is $z_j = (z_{j,1}, z_{j,2}, \ldots, z_{j,l})$. The value $P_{j,p}$ can be considered as the belief of assigning the class $z_{j,p}$ to the test instance by h_j.

In the frame of discernment $\mathcal{Y}$, each classifier h_j can be considered as source of information and its outputs may be regarded as an item of evidence. Regarding the classifier h_j, each class value $\{z_{j,p}\}$ can be represented by the focal element, $\varphi(A_{j,p}, B_{j,p})$ where $A_{j,p} = \{z_{j,p}\}$ and $B_{j,p} = Z_j \setminus \{z_{j,p}\}$, $A_{j,p}$ is the set of labels assigned to one group and $B_{j,p}$ is its complement in Z_j. In a classical *RAKEL* method, the most probable class $z_{j,p}$ is chosen based on the higher value $P_{j,p}$. However, in the *Evidential RAKEL* we consider all probabilities and focal elements, and the item of evidence resulting from each classifier may be given by a mass function over $\mathcal{Y}$ defined by:

$$m_j(A_{j,p}, B_{j,p}) = s.P_{j,p}, \qquad \text{for each } A_{j,p}, B_{j,p} \subseteq Z_j \tag{3.1}$$

$$m_j(\emptyset, \emptyset) = 1 - s.\sum_{p=1}^{l} P_{j,p} = 1 - s, \tag{3.2}$$

Table 3.2: An example of association items of evidence to the outputs of a classifier inside the *Evidential RAKEL* method using 3-label sets ($k = 3$).

Focal elements ($\varphi(A, B)$)	Output probabilities (P_j)	$m_j(A, B)$
$(\emptyset, \{\omega_3, \omega_4, \omega_5\})$	0.0048	0.0043
$(\{\omega_5\}, \{\omega_3, \omega_4\})$	0.0836	0.0753
$(\{\omega_4\}, \{\omega_3, \omega_5\})$	0	0
$(\{\omega_4, \omega_5\}, \{\omega_3\})$	0.0059	0.0053
$(\{\omega_3\}, \{\omega_4, \omega_5\})$	0.0145	0.0130
$(\{\omega_3, \omega_4\}, \{\omega_5\})$	0.8343	0.7508
$(\{\omega_3, \omega_5\}, \{\omega_4\})$	0.0486	0.0437
$(\{\omega_3, \omega_4, \omega_5\}, \emptyset)$	0.0084	0.0076
$(\emptyset, \emptyset)$	-	0.1000

where s is a parameter such that $0 < s < 1$.

For instance, given Linear Discriminant Analysis (LDA) (detailed in Section 4.4.2) as a base classifier h_j, LDA can estimate the posterior probabilities for possible classes in the set $\mathcal{D}_j$ which represented by P_j. Table 3.2 shows the results of applying the LDA method on the Emotions dataset (that will be described in Section 4.4.2), by randomly choosing three labels $Z_j = \{\omega_3, \omega_4, \omega_5\}$ from six. LDA will assign a posterior a probability to each subset of Z_j, called here a focal element. For example, from this table, we can see that the maximum probability is obtained for $\varphi(\{\omega_3, \omega_4\}, \{\omega_5\})$, which is equal to 0.8343. In the *Evidential RAKEL*, this means that we have a degree of belief equal to 0.8343 that the true set of labels of $\mathbf{x}$ contains ω_3 and ω_4, and does not contain ω_5, and we are undecided regarding the other labels $(\omega_1, \omega_2, \omega_6)$. The focal set $\varphi(\emptyset, \{\omega_3, \omega_4, \omega_5\})$ represents the empty set, which is considered as a class in Z_j. The third column in this table computes the mass function corresponding to that item of evidence by fixing s to 0.9.

Once the mass functions induced by the outputs of all classifiers have been calculated, and in order to choose the appropriate set of labels for $\mathbf{x}$, we can combine these mass functions by using a variety of techniques based on evidence theory. In this work, we use Dempster's rule (2.8) for combining elements of evidence:

$$m = \oplus_{j=1}^{m} m_j. \tag{3.3}$$

For making a decision, two techniques can be used: we can calculate the belief for each label separately, or compute the maximum of commonality to predict directly the whole set of labels for the test instance $\mathbf{x}$ (Section 2.3).

3.4.2 Decision Phase

Bel function

After combining all elements of evidence using a combination rule, the *bel* function is calculated. For each class $\omega \in \mathcal{Y}$, we compute the degree of belief $bel(\{\omega\}, \emptyset)$ that the predicted label set Y of $\mathbf{x}$ contains ω, and the degree of belief $bel(\emptyset, \{\omega\})$ that it does not contain ω. A final decision is made by chosen ω if the degree of belief $bel(\{\omega\}, \emptyset)$ is greater than $bel(\emptyset, \{\omega\})$. We then define the multi-label classifier $\mathcal{H}$ as

$$\mathcal{H}(\mathbf{x}) = \{\omega \in \mathcal{Y} \mid bel(\{\omega\}, \emptyset) \geq bel(\emptyset, \{\omega\})\}.$$

Commonality function

An alternative way of making a decision, introduced in [25], is to find the set of labels with the greatest commonality. The commonality function corresponding to the conjunctive sum (3.3) can be obtained by:

$$q \propto \prod_{j=1}^{m} q_j, \tag{3.4}$$

where q_j is the commonality function associated to m_j. These individual commonalities can be expressed for any subset Y of $\mathcal{Y}$ by:

$$q_j(Y) = \begin{cases} 1 - s + s.P_{j,p} & \text{if } Y \in \varphi(A_{j,p}, B_{j,p}) \\ 1 - s & \text{otherwise.} \end{cases} \tag{3.5}$$

The subset $\varphi(A_{j,p}, B_{j,p})$ is the set of all subsets of $\mathcal{Y}$ that include $A_{j,p}$ and are included in $\overline{B}_{j,p}$: $\varphi(A_{j,p}, B_{j,p}) = \{Y \subseteq \mathcal{Y} | A_{j,p} \subseteq Y \subseteq \overline{B}_{j,p}\}$. To simplify calculations, we shall introduce some notations:

$$\begin{aligned} \alpha_j &= 1 - s + s.P_{j,p} \\ \beta_j &= s.P_{j,p} \\ A_{j,p} &= A_j \\ B_{j,p} &= B_j. \end{aligned}$$

We thus have:

$$q_j(Y) = \begin{cases} \alpha_j & \text{if } Y \in \varphi(A_j, B_j) \\ \alpha_j - \beta_j & \text{otherwise,} \end{cases} \tag{3.6}$$

and,

$$q \propto \prod_{j=1}^{m} \alpha_j (\frac{\alpha_j - \beta_j}{\alpha_j})^{1-\delta_j}, \tag{3.7}$$

with

$$\delta_j = \begin{cases} 1 & \text{if } Y \in \varphi(A_{j,p}, B_{j,p}) \\ 0 & \text{otherwise.} \end{cases} \tag{3.8}$$

For each focal element $\varphi(A_j, B_j)$, let us introduce the following notations:

$$a_{jq} = \begin{cases} 1 & \text{if } \omega_q \in A_j \\ 0 & \text{otherwise,} \end{cases} \tag{3.9}$$

and

$$b_{jq} = \begin{cases} 1 & \text{if } \omega_q \in B_j \\ 0 & \text{otherwise.} \end{cases} \tag{3.10}$$

In [25], each subset of labels Y can be represented by a Q-dimensional vector containing integer values (1 or 0) if the corresponding label ω_q is in Y or not. With these notations, the inclusion constraint $A_j \subseteq Y$ may be translated by:

$$\sum_{q=1}^{Q} a_{jq} y_q = \sum_{q=1}^{Q} a_{jq}. \tag{3.11}$$

Similarly, the constraint $Y \subseteq \overline{B}_j$, or, equivalently, $B_j \subseteq \overline{Y}$, may be written as:

$$\sum_{q=1}^{Q} b_{jq}(1 - y_q) = \sum_{q=1}^{Q} b_{jq}. \tag{3.12}$$

Maximizing $q(Y)$ is equivalent to maximizing its logarithm, which is equal to:

$$\begin{aligned} \ln q(Y) &= \sum_{j=1}^{m} [\ln \alpha_j + (1 - \delta_j)(\ln(\alpha_j - \beta_j) - \ln(\alpha_j))] + \text{constant} && (3.13) \\ &= \sum_{j=1}^{m} [\delta_j \ln(\alpha_j) + (1 - \delta_j)(\ln(\alpha_j - \beta_j))] + \text{constant}. && (3.14) \end{aligned}$$

To find the set Y with greatest commonality, the authors in [25] solve the following binary integer programming problem:

$$\min_{y \in \{0,1\}^Q, \delta \in \{0,1\}^m} \sum_{j=1}^{m} \delta_j \ln(\alpha_j - \beta_j), \tag{3.15}$$

subject to the constraints:

$$\begin{cases} \displaystyle\sum_{q=1}^{Q} a_{qj} y_q \geq \delta_j \sum_{q=1}^{Q} a_{qj} \quad \forall\, j = 1 \ldots m \\ \displaystyle\sum_{q=1}^{Q} b_{jq}(1 - y_q) \geq \delta_j \sum_{q=1}^{Q} b_{jq} \quad \forall\, j = 1 \ldots m \end{cases} \tag{3.16}$$

The multi-label classifier $\mathcal{H}$ is defined as

$$\mathcal{H}(\mathbf{x}) = \{Y \subseteq \mathcal{Y} \mid q(Y) = \max q\}.$$

With this technique, we directly obtain the final predicted set of labels for input vector $\mathbf{x}$, but we do not have a score function that calculates the probability of relevance for each label separately.

The full pseudo code for the *Evidential RAKEL* approach is illustrated in Algorithm 1. The parameter s (normalization factor) is an input parameter. It is used with the classifier related to the problem transformation category to get a mass value on the empty set. If $s = 1$, then the source of information h_j is fully reliable. If $s = 0$ the source of information is not reliable at all and all the mass will be assigned to the greatest focal element $\varphi(\emptyset, \emptyset)$. Note that here, we can train not only a single-label classifier but also a multi-label classifier h_j and the outputs may be expressed as degrees of evidence on the existing focal elements. With the *Evidential RAKEL*, we take all outputs instead of the final prediction from each classifier and we combine them in the large frame of discernment. This approach allows us to use all the information provided by each classifier and to capture the relationship between classes. Another advantage of the *Evidential RAKEL* is the reduction of the number of input parameters as compared to the classical *RAKEL* method: the threshold parameter is eliminated, since the decision is automatically made under the belief function framework.

Algorithm 1 *Evidential RAKEL* algorithm.

Input: training dataset $\mathcal{D}$, new instance $\mathbf{x}$, number of labels k randomly chosen from $\mathcal{Y}$, number of models m, classifier h, normalization factor s

Output: predicted set of labels for $\mathbf{x}$ via the *bel* function Y_1, corresponding scoring function f_1, predicted set of labels via the maximum of commonality Y_2

```
1: for $j = 1$ to $m$ do
2:   Identify $Z_j$: the set of $k$ labels randomly chosen from $\mathcal{Y}$;
3:   Identify $\mathcal{D}_j$: the training set corresponding to $Z_j$;
4:   $l = 2^k$;
5:   Learn classifier h by looking for the $l$ subsets $(z_j)$ of $Z_j$ that can be found in $\mathcal{D}_j$
     and their scoring vector $P_j$;
6:   Identify $z_j$ and $P_j$;
7:   for $p = 1$ to $l$ do
8:     $A_{j,p} = \{z_{j,p}\}$;
9:     $B_{j,p} = Z_j \setminus \{z_{j,p}\}$;
10:    $m_j(A_{j,p}, B_{j,p}) = s.P_{j,p}$;
11:  end for
12:  $m_j(\emptyset, \emptyset) = 1 - s$;
13:  Calculate $q_j$ according to Equation 3.5;
14: end for
15: $m = \oplus_{j=1}^{m} m_j$;
16: for $q = 1$ to $Q$ do
17:   $\widehat{f_1}(\mathbf{x}, \omega_q) = bel(\{\omega_q\}, \emptyset)$;
18:   if $\{bel(\{\omega_q\}, \emptyset) \geq bel(\emptyset, \{\omega_q\})\}$ then
19:     $\widehat{Y_1}(q) = +1$;
20:   else
21:     $\widehat{Y_1}(q) = -1$;
22:   end if
23: end for
24: $q \propto \prod_{j=1}^{m} q_j$;
25: $\widehat{Y_2} = \arg\max_{\mathcal{Y}}\{q\}$.
```

3.5 Experimental Evaluation

In this section, we present several experiments with both synthetic and real data in order to study the performances of our approach.

3.5.1 Experiments on Synthetic Data

In this subsection, we use a synthetic dataset to illustrate the *Evidential RAKEL* approach. Our method is compared to the classical *RAKEL* method using a voting process for decision making.

We have generated a two-dimensional dataset with three classes: $\mathcal{Y} = \{\omega_1, \omega_2, \omega_3\}$. The dataset contains 1000 instances drawn from seven Gaussian distributions with the same covariance matrix equal to $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ and means: (-5,-5), (5,-5), (0,5), (0,-5), (-3,1), (3,1) and (0,0). The corresponding labels are respectively $\{\omega_1\}$, $\{\omega_2\}$, $\{\omega_3\}$, $\{\omega_1, \omega_2\}$, $\{\omega_1, \omega_3\}$, $\{\omega_2, \omega_3\}$, and $\{\omega_1, \omega_2, \omega_3\}$. This dataset was randomly divided into training and test datasets with sizes 700 and 300, respectively. Table 3.3 summarizes the distribution of the dataset.

Table 3.3: Distribution of the synthetic data.

Label set	Training instances	Testing instances
$\{\omega_1\}$	154	64
$\{\omega_2\}$	140	72
$\{\omega_3\}$	151	74
$\{\omega_1, \omega_2\}$	60	27
$\{\omega_1, \omega_3\}$	64	25
$\{\omega_2, \omega_3\}$	80	24
$\{\omega_1, \omega_2, \omega_3\}$	51	14

For both methods (*RAKEL* and *Evidential RAKEL*), the number of labels chosen for each model was set to $k = 2$ (2 labels were learnt in each model). The number of models m was fixed to 2 which means that each class will at least be repeated in one model. Note here that the maximum number of models that can be taken is 3. Each model uses the LDA method as base classifier. The threshold parameter for the classical *RAKEL* is set to 0.5. The two sets of labels are $\{\omega_2, \omega_3\}$ and $\{\omega_1, \omega_2\}$.

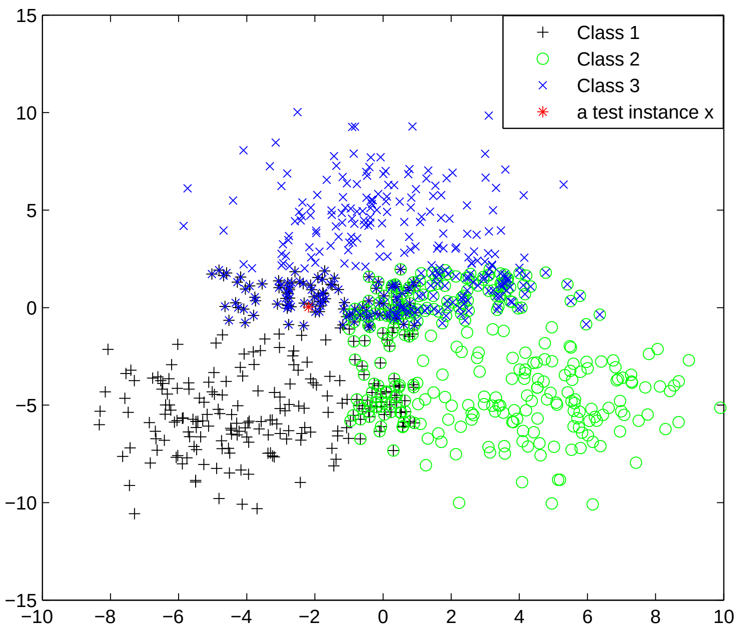


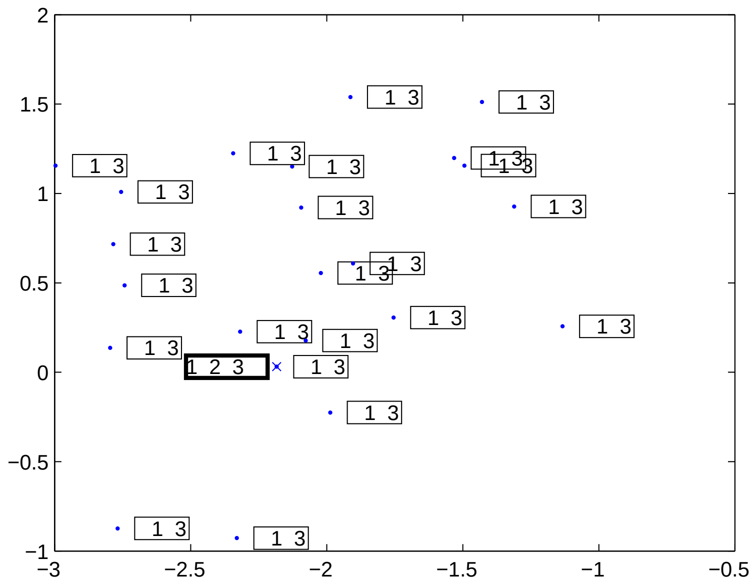
Figure 3.6: Training instances of synthetic data

Hereafter, we show the result of the two methods for a test instance $\mathbf{x}$ which is located in the area of labels $\{\omega_1,\omega_3\}$ (as shown in Figure 3.6). The true label set for $\mathbf{x}$ is $Y = \{\omega_1, \omega_3\}$. Figure 3.7 shows a part of the region space for $\mathbf{x}$, and the estimated label sets for this instance by using the *RAKEL* and *Evidential RAKEL* methods.

To estimate the predicted label set for $\mathbf{x}$ using the classical *RAKEL*, we look for the outputs of classifiers h_1 and h_2. We obtain the following probabilities from h_1:

$$\begin{aligned}
\mathrm{P}(\overline{\omega_2}, \overline{\omega_3}) &= 0.0412\\
\mathrm{P}(\overline{\omega_2}, \omega_3) &= 0.4727\\
\mathrm{P}(\omega_2, \overline{\omega_3}) &= 0.0022\\
\mathrm{P}(\omega_2, \omega_3) &= 0.4839,
\end{aligned}$$

the estimated label set for $\mathbf{x}$ given by classifier h_1 is $\{\omega_2, \omega_3\}$. It is chosen by selecting



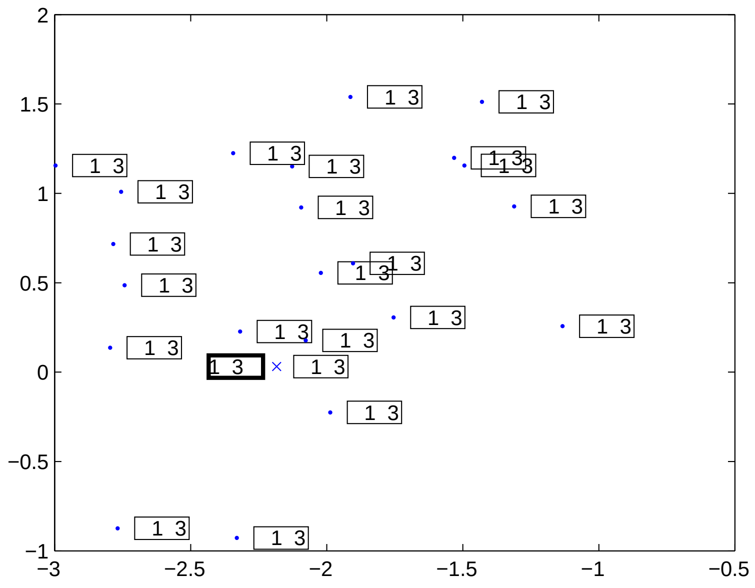


Figure 3.7: Estimated label set (in bold) for a test instance using the classical *RAKEL* (top) and *Evidential RAKEL* (bottom) methods.

the label set having the highest probability. From classifier h_2, we get:

$$\begin{aligned}\mathrm{P}(\overline{\omega_1}, \overline{\omega_2}) &= 0.1835\\ \mathrm{P}(\overline{\omega_1}, \omega_2) &= 0.0024\\ \mathrm{P}(\omega_1, \overline{\omega_2}) &= 0.3616\\ \mathrm{P}(\omega_1, \omega_2) &= 0.4525.\end{aligned}$$

Thus, the estimated label set for $\mathbf{x}$ given by classifier h_2 is $\{\omega_1, \omega_2\}$. To calculate the final predictions, the outputs of these two classifiers are averaged, as seen in Table 3.4, and we obtain $\widehat{Y} = \{\omega_1, \omega_2, \omega_3\}$ as final output.

Table 3.4: Results for a test instance in the synthetic dataset with the classical *RAKEL* approach.

classifier	2-label sets	ω_1	ω_2	ω_3
h_1	$\{\omega_2, \omega_3\}$	0	1	1
h_2	$\{\omega_1, \omega_2\}$	1	1	0
	average votes	1/1	2/2	1/1
	final prediction	1	1	1

By applying the *Evidential RAKEL*, we can associate a mass function to the outputs of each classifier. The mass function resulting from classifier h_1 is:

$$\begin{aligned}m_1(\emptyset, \{\omega_2, \omega_3\}) &= 0.0371\\ m_1(\{\omega_3\}, \{\omega_2\}) &= 0.4251\\ m_1(\{\omega_2\}, \{\omega_3\}) &= 0.0020\\ m_1(\{\omega_2, \omega_3\}, \emptyset) &= 0.4355\\ m_1(\emptyset, \emptyset) &= 0.1000.\end{aligned}$$

Equivalently, the mass function induced from the outputs of classifier h_2 is:

$$\begin{aligned}m_2(\emptyset, \{\omega_1, \omega_2\}) &= 0.1652\\ m_2(\{\omega_2\}, \{\omega_1\}) &= 0.0022\\ m_2(\{\omega_1\}, \{\omega_2\}) &= 0.3254\\ m_2(\{\omega_1, \omega_2\}, \emptyset) &= 0.4072\\ m_2(\emptyset, \emptyset) &= 0.1000.\end{aligned}$$

Table 3.5: Computation of $m_1 \textcircled{\cap} m_2$ resulting from the two mass functions in the synthetic dataset example.

	$(\emptyset, \{\omega_2, \omega_3\})$ 0.0371	$(\{\omega_3\}, \{\omega_2\})$ 0.4251	$(\{\omega_2\}, \{\omega_3\})$ 0.0020	$(\{\omega_2, \omega_3\}, \emptyset)$ 0.4355	$(\emptyset, \emptyset)$ 0.1000
$(\emptyset, \{\omega_1, \omega_2\})$ 0.1652	$(\emptyset, \{\omega_1, \omega_2, \omega_3\})$ 0.1652×0.0371	$(\{\omega_3\}, \{\omega_1, \omega_2\})$ 0.1652×0.4251	$\emptyset_{\mathcal{Y}}$ 0.1652×0.0020	$\emptyset_{\mathcal{Y}}$ 0.1652×0.4355	$(\emptyset, \{\omega_1, \omega_2\})$ 0.1652×0.1000
$(\{\omega_2\}, \{\omega_1\})$ 0.0022	$\emptyset_{\mathcal{Y}}$ 0.0022×0.0371	$\emptyset_{\mathcal{Y}}$ 0.0022×0.4251	$(\{\omega_2\}, \{\omega_1, \omega_3\})$ 0.0022×0.0020	$(\{\omega_2, \omega_3\}, \{\omega_1\})$ 0.0022×0.4355	$(\{\omega_2\}, \{\omega_1\})$ 0.0022×0.1000
$(\{\omega_1\}, \{\omega_2\})$ 0.3254	$(\{\omega_1\}, \{\omega_2, \omega_3\})$ 0.3254×0.0371	$(\{\omega_1, \omega_3\}, \{\omega_2\})$ 0.3254×0.4251	$\emptyset_{\mathcal{Y}}$ 0.3254×0.0020	$\emptyset_{\mathcal{Y}}$ 0.3254×0.4355	$(\{\omega_1\}, \{\omega_2\})$ 0.3254×0.1000
$(\{\omega_1, \omega_2\}, \emptyset)$ 0.4072	$\emptyset_{\mathcal{Y}}$ 0.4072×0.0371	$\emptyset_{\mathcal{Y}}$ 0.4072×0.4251	$(\{\omega_1, \omega_2\}, \{\omega_3\})$ 0.4072×0.0020	$(\{\omega_1, \omega_2, \omega_3\}, \emptyset)$ 0.4072×0.4355	$(\{\omega_1, \omega_2\}, \emptyset)$ 0.4072×0.1000
$(\emptyset, \emptyset)$ 0.1000	$(\emptyset, \{\omega_2, \omega_3\})$ 0.1000×0.0371	$(\{\omega_3\}, \{\omega_2\})$ 0.1000×0.4251	$(\{\omega_2\}, \{\omega_3\})$ 0.1000×0.0020	$(\{\omega_2, \omega_3\}, \emptyset)$ 0.1000×0.4355	$(\emptyset, \emptyset)$ 0.1000×0.1000

Table 3.6: Computation of the *bel* function for the synthetic dataset example.

	$bel(\{\omega_q\}, \emptyset)$	$bel(\emptyset, \{\omega_q\})$
$q = 1$	0.6744	0.1578
$q = 2$	0.4426	0.5406
$q = 3$	0.7938	0.0385

We combine these two mass functions using the Dempster's rule (2.8) as shown in Table 3.5.

In order to make a decision about the final predicted label set for $\mathbf{x}$, we need to compute the *bel* function and to compare the two quantities presented in Equation 3.4.2, for each $q \in \{1, 2, 3\}$ as shown in Table 3.6. Thus, we get $\widehat{Y}' = \{\omega_1, \omega_3\}$ as final label set. We can see that, by using the LDA classifier with the *Evidential RAKEL*, we can estimate the true label set for $\mathbf{x}$, while by using the classical *RAKEL*, the estimated label set is erroneous. The fact that the two results are different is not surprising if we observe the outputs of the two classifiers. By examining the outputs of classifier h_1, we observe that the highest probability (0.4839) is assigned to the label set $\{\omega_2, \omega_3\}$, while the second largest probability (0.4727) is assigned to class $\{\overline{\omega_2}, \omega_3\}$; this latter value can be interpreted as a degree of belief that $\mathbf{x}$ belongs to $\{\omega_3\}$ and does not belong to $\{\omega_2\}$. Regarding the outputs of classifier h_2, the highest probability (0.4525) is assigned to $\{\omega_1, \omega_2\}$ while the second largest probability (0.3616) is assigned to $\{\omega_1, \overline{\omega_2}\}$, which means that we have some degree of belief that $\mathbf{x}$ does not belong to ω_2. Using the *Evidential RAKEL*, all these degrees of belief are considered when combining the two mass functions and making the decision. However, the classical *RAKEL* may not be able to take this information in consideration when determining the label set of $\mathbf{x}$.

Thus, we can conclude that the *Evidential RAKEL* method can take into account all the information provided by a classifier, and may correct some lack of information when calculating the label set for a test instance.

3.5.2 Experiments on Real-World Data

To evaluate the performance of our algorithm on real data, we carried out some experiments with three multi-labelled datasets from the domain of multimedia (music and image) classification. We applied our approach with three existing classification methods: Linear Discriminant Analysis (LDA), The Classification And Regression Tree (CART), and the Evidential k-nearest neighbor for multi-label classification (EMLkNN). Before presenting our experimental results, we first shortly describe the benchmark multi-label datasets. We then give a short overview of the evaluation measures for multi-label classification used in our experiments. Next, we explain the base classifiers used in our approach and the parameter setting of these methods.

Datasets

Our method was experimented using the Emotions, Scene and Image datasets [1].

– *Emotions dataset.* This dataset consists of 593 songs labelled by experts according to the emotions they generated. Each piece of music is described by 8 rhythmic features and 64 timbre features, and can be annotated with the following emotions: *amazed-surprised, happy-pleased, relaxing-calm, quiet-still, sad-lonely and angry-fearful.* The average number of labels for each song is 1.869, and the number of distinct label sets is equal to 27 [91].

– *Scene dataset.* The Scene dataset consists of 2407 natural scene images. There are six different semantic classes: *beach, sunset, foliage, field, mountain, urban.* Spatial color moments are used as features. Each image is divided into 49 blocks using 7×7 grid. The mean and variance of each band are computed corresponding to a low-resolution image and to computationally inexpensive texture features, respectively. Each image is then described by $49 \times 2 \times 3$ features. The average number of labels for each picture is 1.074, and the number of distinct label sets is equal to 15 [8].

1. http://mulan.sourceforge.net/datasets.html

– *Image dataset.* This dataset contains 2000 natural scene images belonging to the following classes: *desert, mountains, sea, sunset, trees.* Each image is represented by a feature vector using the same method employed for the Scene dataset [115]. The average number of labels per instance is 1.24, and the number of distinct set of labels is equal to 20.

Evaluation measures

As previously discussed in Chapter 1, performance evaluation of multi-label classifiers requires different measures than those used in traditional classification. In our experiments, we use the prediction-based measures (Hamming loss, Accuracy, Precision, Recall and F1-measure).

Base classifiers

In our experiments, we select two types of existing base classifiers: the LDA and the CART decision tree learning algorithms chosen from the traditional single-label classification literature, and the EMLkNN method from the category of problem adaptation methods for multi-label learning.

Linear Discriminant Analysis (LDA) is used to generate a set of linear discriminant functions, one for each class. This method is based on the assumption that the feature vector in each class has a multivariate normal distribution with equal covariance matrix. In order to make a decision for an unseen instance $\mathbf{x}$, LDA estimates the posterior probability for each group of the set Z_j. The Classification And Regression Tree (CART) method uses the set of training data with the predefined classes for building trees. The CART algorithm constructs binary trees, namely each internal node has exactly two outgoing nodes. It searches for all possible variables and all possible values in order to find the best split. The process is repeated for each of the resulting data fragments. Splitting may stop when CART detects no further gain can be made [9]. The Evidential k-nearest neighbor for multi-label classification (EMLkNN) is based on the evidential k-NN rule. This method was discussed in Section 2.4.

Parameter setting

In our experiments, we used the following parameters:

– For the classical *RAKEL* and *Evidential RAKEL* approaches, we set the number of label sets to 3 ($k = 3$) and the number of models ranges from Q to $2 * Q$.
– For the EMLkNN method, we used the Euclidian distance, the parameter γ was fixed to 0.1, and the number of nearest neighbors was set to 10.
– For the methods belonging to problem transformation category (LDA and CART), we set the normalization factor s to 0.9.
– For the classical *RAKEL* approach, in the voting process, the threshold parameter was set to 0.5 ($\mathbf{t} = 0.5$) according to [95].

Results and discussion

Our approach was compared to the classical *RAKEL* using two types of decision: the voting process with threshold equal to 0.5 (Vote), and the threshold selected via some statistics of multi-labelled datasets. This last method, known as the Ensemble Multi-Label learning approach (EML), uses the difference between the label cardinality of the training dataset ($\mathcal{D}$) and the predictions made on the testing set ($\mathcal{S}$) to select the corresponding threshold in the making decision step [87]:

$$\mathbf{t} = \arg \min_{\mathbf{t} \in \{0:0.1:1\}} |LCard(\mathcal{D}) - LCard(\mathcal{H}(\mathcal{S}))|,$$

where the label cardinality measure, LCard(X), is the average number of labels per instance. All experiments were conducted with 10-fold cross-validation.

Due to randomization of label space, results are very sensitive to the selected combination of labels. To deal with this negative aspect, we grouped results in batches of 10 classifiers calculated for the same value of m, and we computed the average and the standard deviation. The performance of all methods are shown in Tables 3.7-3.15. In these tables, the rank of each method is given, and best value on each evaluation criterion is highlighted in bold letters. To statistically measure the significance of performance difference, two-tailed paired t-tests at 5% significance level were performed between the *Evidential RAKEL* and the *RAKEL* approaches.

The presented experimental results can be summarized as follows:

– Tables 3.7, 3.10 and 3.13 indicate that our method *Evidential RAKEL LDA* outperforms in average the original *RAKEL* with the two values of threshold (0.5 and difference of LCard). We note that there is no clear difference in our method between the two rules of decision, MaxCom and Bel.

- From the results in Tables 3.8, 3.11 and 3.14, we can say that our *Evidential RAKEL DT* yields good performances and it is competitive with the two versions of decision on all evaluation metrics except the *recall* criterion.
- We remark that for the two classifiers (LDA and DT), the classical *RAKEL* outperforms our method on the *recall* criterion. This can be explained by the fact that the classical *RAKEL* method tends to classify most examples as positive with respect to all existing classes. This also explains why our method is superior to classical *RAKEL* by taking the other metrics as a reference.
- We note that we cannot apply the EMLkNN method, with the classical *RAKEL*, since with the voting process we need to have a final decision in each classifier, to average all results and output the final decision. In this case, the different mass functions given on Z_j should be transferred to $\mathcal{Y}$ by conditioning before combining. Tables 3.9, 3.12 and 3.15 show the comparison between the two rules of decision for the *Evidential RAKEL EMLkNN*.

We have to say that one of the major drawback of the *Evidential RAKEL* is the space complexity: we have to manipulate 3^k subsets in each classifier, which makes a total of $m * 3^k$ subsets. The complexity thus increases with Q, as m is going from Q to $2 * Q$.

Table 3.7: Experimental results (mean±std) on the Emotions dataset using the ADL classifier.

	MaxCom	Bel	EML	Vote
Hamming loss$^-$	0.2394 ± 0.0050(2)	0.2365 ± 0.0050•(1)	0.2487 ± 0.0014•(3)	0.2524 ± 0.0028•(4)
Accuracy$^+$	0.5142 ± 0.0086(1)	0.5137 ± 0.0092○(2)	0.4992 ± 0.0049•(4)	0.5070 ± 0.0034•(3)
Precision$^+$	0.6328 ± 0.0094(2)	0.6356 ± 0.0088○(1)	0.6187 ± 0.0049•(3)	0.6127 ± 0.0038•(4)
Recall$^+$	0.6303 ± 0.0084(2)	0.6220 ± 0.0078•(4)	0.6227 ± 0.0141•(3)	0.6502 ± 0.0056•(1)
F1$^+$	0.6031 ± 0.0087(1)	0.6012 ± 0.0087○(3)	0.5910 ± 0.0056•(4)	0.6015 ± 0.0033○(2)

+(-): the higher (smaller) the value, the better the performance.

•(○): statistically significant (non-significant) difference of performance between the *Evidential RAKEL-MaxCom* and the compared algorithm, based on two-tailed paired t-test at 5% significance.

Table 3.8: Experimental results (mean±std) on the Emotions dataset using the DT classifier.

	MaxCom	Bel	EML	Vote
Hamming loss$^-$	0.2583 ± 0.0069(1)	0.2596 ± 0.0047○(2)	0.2748 ± 0.0062●(4)	0.2689 ± 0.0088●(3)
Accuracy$^+$	0.4577 ± 0.0129(1)	0.4503 ± 0.0106●(2)	0.4436 ± 0.0118●(3)	0.4275 ± 0.0066●(4)
Precision$^+$	0.5796 ± 0.0128(1)	0.5750 ± 0.0109●(2)	0.5554 ± 0.0104●(4)	0.5568 ± 0.0146●(3)
Recall$^+$	0.5427 ± 0.0092(2)	0.5351 ± 0.0080●(3)	0.5672 ± 0.0165●(1)	0.5203 ± 0.0104●(4)
F1$^+$	0.5350 ± 0.0115(1)	0.5284 ± 0.0097●(3)	0.5322 ± 0.0123○(2)	0.5094 ± 0.0050●(4)

+(-): the higher (smaller) the value, the better the performance.

●(○): statistically significant (non-significant) difference of performance between the *Evidential RAKEL-MaxCom* and the compared algorithm, based on two-tailed paired t-test at 5% significance.

Table 3.9: Experimental results (mean±std) on the Emotions dataset using the EMLkNN classifier.

	MaxCom	Bel
Hamming loss$^-$	0.2181 ± 0.0014(1)	0.2438 ± 0.0300○(2)
Accuracy$^+$	0.5465 ± 0.0020(1)	0.5345 ± 0.0133●(2)
Precision$^+$	0.6510 ± 0.0014(1)	0.6352 ± 0.0187○(2)
Recall$^+$	0.6391 ± 0.0041(2)	0.6434 ± 0.0081●(1)
F1$^+$	0.6216 ± 0.0020(1)	0.6118 ± 0.0115●(2)

+(-): the higher (smaller) the value, the better the performance.

●(○): statistically significant (non-significant) difference of performance between the *Evidential RAKEL-MaxCom* and the compared algorithm, based on two-tailed paired t-test at 5% significance.

Table 3.10: Experimental results (mean±std) on the Scene dataset using the ADL classifier.

	MaxCom	Bel	EML	Vote
Hamming loss$^-$	0.1351 ± 0.0024(3)	0.1343 ± 0.0021○(2)	0.1337 ± 0.0015●(1)	0.1444 ± 0.0020●(4)
Accuracy$^+$	0.5981 ± 0.0053(1)	0.5977 ± 0.0054○(2)	0.5899 ± 0.0031●(3)	0.5810 ± 0.0028●(4)
Precision$^+$	0.6276 ± 0.0058(2)	0.6277 ± 0.0059○(1)	0.6181 ± 0.0034●(3)	0.6057 ± 0.0032●(4)
Recall$^+$	0.6634 ± 0.0013(2)	0.6516 ± 0.0024●(3)	0.6493 ± 0.0071●(4)	0.6858 ± 0.0046●(1)
F1$^+$	0.6299 ± 0.0041(1)	0.6258 ± 0.0032●(2)	0.6194 ± 0.0034●(4)	0.6244 ± 0.0013●(3)

+(-): the higher (smaller) the value, the better the performance.

●(○): statistically significant (non-significant) difference of performance between the *Evidential RAKEL-MaxCom* and the compared algorithm, based on two-tailed paired t-test at 5% significance.

3.6 Conclusion

In multi-label classification, it is customary to transform multi-labelled instances to the domain of single-label and to classify these instances using LP classifiers. However,

Table 3.11: Experimental results (mean±std) on the Scene dataset using the DT classifier.

	MaxCom	Bel	EML	Vote
Hamming loss$^-$	0.1179 ± 0.0079(2)	0.1153 ± 0.0060●(1)	0.1282 ± 0.0080●(4)	0.1243 ± 0.0093●(3)
Accuracy$^+$	0.5811 ± 0.0161(2)	0.5833 ± 0.0158○(1)	0.5753 ± 0.0213○(3)	0.5533 ± 0.0101●(4)
Precision$^+$	0.6087 ± 0.0174(2)	0.6112 ± 0.0166○(1)	0.5996 ± 0.0223●(3)	0.5783 ± 0.0111●(4)
Recall$^+$	0.6095 ± 0.0046(2)	0.6086 ± 0.0062○(3)	0.6453 ± 0.0130●(1)	0.6005 ± 0.0116●(4)
F1$^+$	0.5999 ± 0.0126(3)	0.6012 ± 0.0128○(1)	0.6069 ± 0.0189●(2)	0.5776 ± 0.0061●(4)

+(-): the higher (smaller) the value, the better the performance.
●(○): statistically significant (non-significant) difference of performance between the *Evidential RAKEL-MaxCom* and the compared algorithm, based on two-tailed paired t-test at 5% significance.

Table 3.12: Experimental results (mean±std) on the Scene dataset using the EMLkNN classifier.

	MaxCom	Bel
Hamming loss$^-$	0.1050 ± 0.0003(2)	0.1033 ± 0.0004●(1)
Accuracy$^+$	0.6651 ± 0.0027(1)	0.6586 ± 0.0030●(2)
Precision$^+$	0.7012 ± 0.0029(1)	0.6938 ± 0.0032●(2)
Recall$^+$	0.6663 ± 0.0021(1)	0.6605 ± 0.0028●(2)
F1$^+$	0.6775 ± 0.0026(1)	0.6709 ± 0.0030●(2)

+(-): the higher (smaller) the value, the better the performance.
●(○): statistically significant (non-significant) difference of performance between the *Evidential RAKEL-MaxCom* and the compared algorithm, based on two-tailed paired t-test at 5% significance.

this transformation is challenged by loss of information due to the fact that we consider only a subset of labels in each base classifier. In this chapter, we reviewed the *RAKEL* method, which deals with these problems but has some limitations; moreover, this method has several parameters that need to be tuned a priori. This chapter described a solution for these problems based on the theory of evidence.

Evidential RAKEL is an efficient multi-label classification method. The idea is to address the limitation of the *RAKEL* approach while still taking into account the correlation between labels. Thus, *Evidential RAKEL* reduces the number of parameters of the *RAKEL* method by allowing us to circumvent the need of specifying a threshold parameter. We illustrated our method on synthetic and real datasets and we compared the results with those of the classical *RAKEL* method. Experimental results demonstrated that *Evidential RAKEL* performs significantly better than the *RAKEL* approach for all methods (ADL, CART and EMLkNN) with respect to all evaluation metrics and on the different datasets.

Table 3.13: Experimental results (mean±std) on the Image dataset using the ADL classifier.

	EML*k*NN		Rank-SVM	
	MaxCom	Bel	EML	Vote
Hamming loss$^-$	0.2505 ± 0.0018(3)	0.2440 ± 0.0011●(2)	0.2411 ± 0.0046●(1)	0.2669 ± 0.0044●(4)
Accuracy$^+$	0.4727 ± 0.0037(2)	0.4766 ± 0.0039●(1)	0.4669 ± 0.0033●(3)	0.4625 ± 0.0038●(4)
Precision$^+$	0.5229 ± 0.0039(2)	0.5272 ± 0.0045●(1)	0.5209 ± 0.0038○(4)	0.5092 ± 0.0050●(3)
Recall$^+$	0.5651 ± 0.0031(2)	0.5586 ± 0.0032●(3)	0.5473 ± 0.0142●(4)	0.6024 ± 0.0057●(1)
F1$^+$	0.5211 ± 0.0033(3)	0.5219 ± 0.0030○(2)	0.5126 ± 0.0062●(4)	0.5240 ± 0.0029○(1)

+(-): the higher (smaller) the value, the better the performance.
●(○): statistically significant (non-significant) difference of performance between the *Evidential RAKEL-MaxCom* and the compared algorithm, based on two-tailed paired t-test at 5% significance.

Table 3.14: Experimental results (mean±std) on the Image dataset using the DT classifier.

	EML*k*NN		Rank-SVM	
	MaxCom	Bel	EML	Vote
Hamming loss$^-$	0.2583 ± 0.0134(2)	0.2530 ± 0.0111○(1)	0.2875 ± 0.0100●(4)	0.2812 ± 0.0132●(3)
Accuracy$^+$	0.3935 ± 0.0189(1)	0.3864 ± 0.0149○(2)	0.3227 ± 0.0181●(3)	0.3042 ± 0.0207●(4)
Precision$^+$	0.4515 ± 0.0222(1)	0.4460 ± 0.0176○(2)	0.3729 ± 0.0139●(3)	0.3569 ± 0.0181●(4)
Recall$^+$	0.4634 ± 0.0101(1)	0.4470 ± 0.0092●(2)	0.4205 ± 0.0432●(3)	0.3826 ± 0.0434●(4)
F1$^+$	0.4377 ± 0.0166(1)	0.4279 ± 0.0133○(2)	0.3732 ± 0.0246●(3)	0.3492 ± 0.0270●(4)

+(-): the higher (smaller) the value, the better the performance.
●(○): statistically significant (non-significant) difference of performance between the *Evidential RAKEL-MaxCom* and the compared algorithm, based on two-tailed paired t-test at 5% significance.

Table 3.15: Experimental results (mean±std) on the Image dataset using the EML*k*NN classifier.

	EML*k*NN	
	MaxCom	Bel
Hamming loss$^-$	0.2002 ± 0.0010(2)	0.1954 ± 0.0014●(1)
Accuracy$^+$	0.5354 ± 0.0017(1)	0.5338 ± 0.0035○(2)
Precision$^+$	0.6088 ± 0.0021(1)	0.6051 ± 0.0038●(2)
Recall$^+$	0.5491 ± 0.0019(1)	0.5490 ± 0.0036○(2)
F1$^+$	0.5643 ± 0.0017(1)	0.5626 ± 0.0036○(2)

+(-): the higher (smaller) the value, the better the performance.
●(○): statistically significant (non-significant) difference of performance between the *Evidential RAKEL-MaxCom* and the compared algorithm, based on two-tailed paired t-test at 5% significance.

Chapter 4

Editing multi-labelled data

4.1 Introduction

In classification problems, performance of algorithms depends greatly on the quality of the learning data. Real world data often suffer from noisy or erroneous instances due to several problems, like errors in the input vectors or in their labels. To cope with this problem in the framework of single-label learning, several methods based on data reduction have been introduced. These techniques are usually based on prototype selection [11][36][73][97].

Prototype selection methods are usually applied to remove erroneous or redundant instances from the training dataset [36][56][65]. These methods are widely used with the traditional nearest neighbor rule due to their simplicity and effectiveness. In addition to improving classification accuracy for unseen instances, using prototypes dramatically decreases storage and classification-time costs.

However, despite extensive works in multi-label learning [22][60][71][94][109][111], there is a lack of methods for improving the quality of multi-labelled training instances. This fact motivated us to study this problem in the framework on multi-label learning. We developed an original method based on a prototype selection using the nearest neighbor rule and a local criterion, in order to purify training dataset and improve the performances of multi-label classification algorithms.

This chapter is organized as follows. Background notions on the nearest neighbor rule in the classical single-label framework and some related techniques for prototype selection will first be recalled in Section 4.2. Our approach will then be introduced in

Section 4.3. Section 4.4 reports the experimental evaluation of the presented methods on synthetic and real-world datasets. Finally, our contribution will be summarized in Section 4.5.

4.2 Related works on prototype selection for single-labelled data

The problem of noise handling has received considerable attention in the literature on machine learning. Seeking to start with something relatively simple, scientists have focused on the nearest neighbor classifier considered as one of the most well-known technique in machine learning and data mining due to its simplicity and effectiveness. Given a training set of single-labelled data, the idea is to select an optimal set of training instances, known as prototypes, in order to maximize the performances of the Nearest Neighbor (NN) classifier and/or to minimize the computing time of this classifier [**?**]. Later, the idea of selecting "good" instances has also been applied to other types of classifiers [11]. In this section, we will rapidly review the nearest neighbor rule, and give a definition and summary of works related to prototype selection methods for the NN rule and other classification methods.

Nearest Neighbor classification

The Nearest Neighbor rule [20] is a well-known and non-parametric decision procedure for machine learning and data mining tasks. It has been considered as one of the most effective algorithms in machine learning, and one of the top ten methods in data mining [36][107]. In traditional supervised learning, this rule assigns to an unseen sample $\mathbf{x}$, the class of the nearest training instance according to some distance metric. The voting k-nearest neighbor rule (k-NN), with $k > 1$, is a generalization of the NN approach where the predicted class of $\mathbf{x}$ is set as equal to the class represented a majority of its k nearest neighbors in the training set.

However, the k-NN rule suffers from several problems such as large storage requirements, high computational complexity in the operational phase, and low tolerance to noise due to considering all instances as relevant while the training set may contain noisy or mislabelled examples. Different techniques have been proposed in the literature to alleviate these problems. One technique, known as prototype selection, consists of selecting an appropriate subset of the training data that yields a similar or even higher classification accuracy. Prototype selection methods can be categorized into three different families. First, *edition methods* eliminate noisy instances from the original training set in order to improve classification accuracy. Second, *condensation methods* select a sufficiently small subset of training instances which lead to the same performance of the single nearest neighbor rule (1-NN), by removing instances that will not affect classification accuracy. Finally, *hybrid methods* select a small subset of training instances that incorporates the goals of these two previous methods [10][36].

In the following, we will provide more details about prototype selection in single-label learning. Figure 4.1 shows a map of the main methods proposed in the literature [10][35][36].

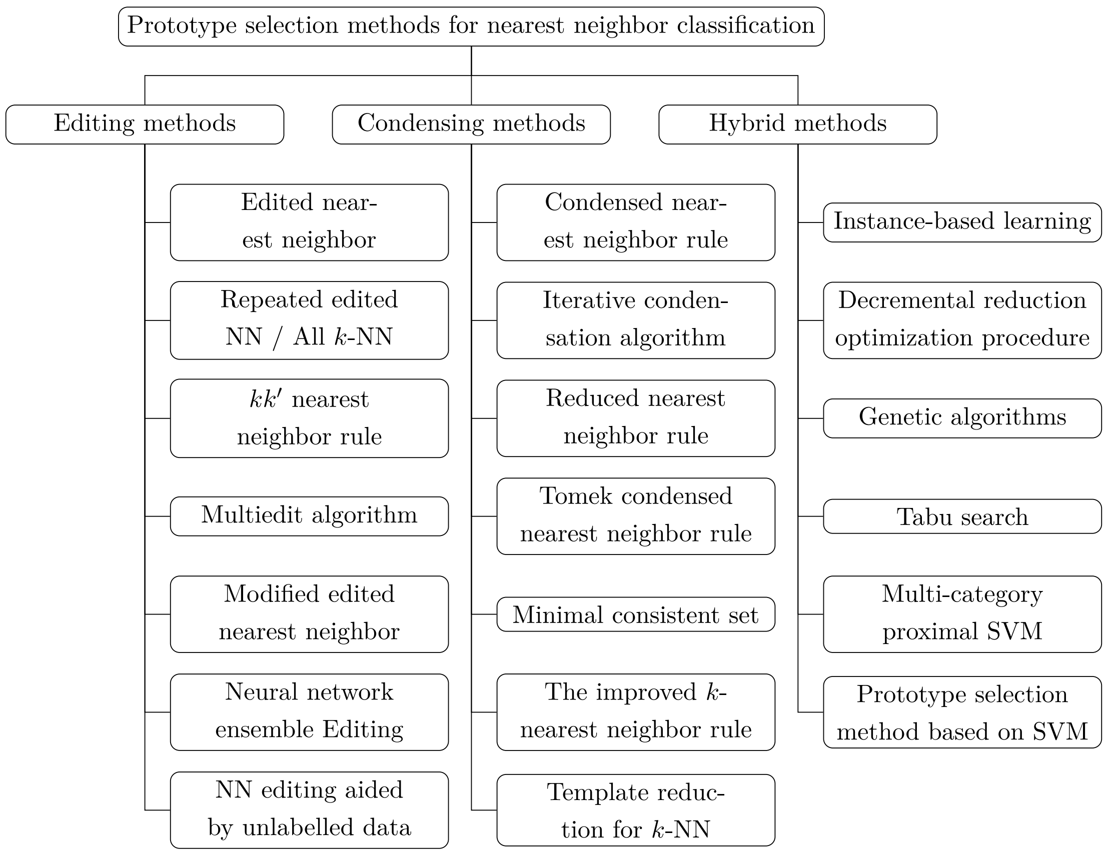

Prototype selection methods for nearest neighbor classification
Editing methods
Condensing methods
Hybrid methods
Edited nearest neighbor
Repeated edited NN / All k-NN
kk' nearest neighbor rule
Multiedit algorithm
Modified edited nearest neighbor
Neural network ensemble Editing
NN editing aided by unlabelled data
Condensed nearest neighbor rule
Iterative condensation algorithm
Reduced nearest neighbor rule
Tomek condensed nearest neighbor rule
Minimal consistent set
The improved k-nearest neighbor rule
Template reduction for k-NN
Instance-based learning
Decremental reduction optimization procedure
Genetic algorithms
Tabu search
Multi-category proximal SVM
Prototype selection method based on SVM

Figure 4.1: Prototype selection methods for nearest neighbor classification

Editing methods

Editing methods process the training data by removing border and noisy instances or making other necessary cleaning, with the aim of improving classification accuracy of learning algorithms on test data. Below we first review some algorithms related to the editing approach for the nearest neighbor rule. We then provide an overview of editing methods that are not specific to the nearest neighbor rule but contribute to output better quality data for any learning algorithm.

Wilson proposed the first editing rule [105], called Edited Nearest Neighbor (ENN), to improve the performance of the 1-NN rule. This method can be described in the following manner. Each instance in the training set is classified using the k-NN rule and it is marked if its predicted class does not agree with the true class. Edition is achieved by deleting all misclassified instances at once. After, any input sample is classified using the 1-NN rule with the remaining instances. Experiments with the editing rule were reported by Tomek who proposed two variants of the ENN rule: RENN and All k-NN [90]. The Repeated Edited Nearest Neighbor (RENN) rule repeats the ENN algorithm until a stable set is obtained where no more samples are edited out. The All k-NN applies iteratively the ENN algorithm with the i-NN rule where i is going from 1 to k.

In [52], the generalized editing procedure based on the kk'-NN rule was introduced. The purpose of this procedure was two-fold: improving the level of performance of the ENN algorithm and reducing the proportion of deleted samples. Based on the class of a majority of k' instances from a group of k nearest samples to an instance $\mathbf{x}$, the group of k samples is either deleted or relabelled as belonging to the majority class. The 1-NN is then used on the edited set to classify an input instance. In [27], the authors proposed the well-known Multiedit algorithm, which randomly breaks the initial training set into different subsets. In each subset, every instance is classified using the 1-NN rule with the instances in the next subset. Misclassified instances are discarded. The remaining instances constitute a new set and the algorithm is iteratively repeated until no more instances are edited out.

In [44], a Modified Edited k-NN rule (MEKNN) was proposed. According to this rule, a sample $\mathbf{x}$ is deleted from the initial set if its class does agree with the class of its k nearest neighbors and their tying instances (tying instances are those in the training set that are at the same distance to $\mathbf{x}$ as its furthest neighbor). In addition, this method introduces a fixed number of pairs (k, k'). k is the number of neighbors to make the edition process and k' is employed to classify any new instance in the obtained edited set. The goal was to obtain the optimal pairs of k and k' to employ the final editing reference set.

In [46], the authors proposed an ensemble of neural networks to edit the training set of k-NN classifiers. The Neural Network Ensemble Editing (NNEE) algorithm uses bagging predictors to construct a neural network ensemble from the original training dataset. The class label of training examples are replaced by the label predicted by the neural network ensemble. This algorithm works with two parameters, i.e., the number of neural networks used in the neural network ensemble, and the number of hidden units in the networks.

Another method for nearest neighbor editing was proposed in [42]. This method uses the concept of semi-supervised learning and edits the training instances by using the whole dataset including: labelled and unlabelled instances. The proposed method, called NNEAUD (Nearest Neighbor Editing Aided by Unlabelled Data), consists of two steps: labels are first predicted for unlabelled instances, and the augmented dataset is then used in data editing. The NNEAUD method uses ENN, RENN, and All$k$NN algorithms with unlabelled data to edit the training instances.

The literature survey indicates that only a few papers have so far dealt with general editing methods that can be applied to any dataset before feeding it to a learning algorithm. In [11] [101], the authors proposed an approach to identify and eliminate mislabelled instances from the training set for supervised learning. The idea was to use a set of learning algorithms to create classifiers that serve as filters for the training data. Instances that are well classified by the filter are selected to obtain the reduced dataset.

Condensing methods

Condensing methods try to find a significantly reduced subset that leads to the same performance as the initial training dataset while reducing the computation time when classifying new instances. The idea behind these methods is to retain instances that are near the decision boundaries and remove instances that are far away from the decision boundaries. Below is a summary of some condensing methods.

Hart was one of the first who proposed a condensing algorithm, named Condensed Nearest Neighbor Rule (CNN) [**?**][3] in 1968, for reducing the given training dataset into a consistent subset of instances that correctly classifies all original instances with the nearest neighbor rule. CNN begins by randomly selecting a representative prototype from each class to form the initial condensed subset. Using this subset with the NN rule, each misclassified instance is iteratively added to the condensed subset to ensure its correct classification. This process is repeated until there are no more additions to the condensed neighbors subset. One drawback of this method is its sensitivity to the initial ordering of the training dataset. Furthermore, it is suitable only for the 1-NN rule. To tackle this problem, Swonger proposed in [86] the Iterative Condensation Algorithm (ICA), which can either add or delete instances from the condensed set. ICA introduced the notion of *margin of classification*, defined as the difference between the distance of $\mathbf{x}$ to the closest prototype of the correct class, minus the distance to the closest prototype of an incorrect class, which is negative for incorrectly classified instances and positive for correctly classified instances. ICA starts with a prototype set containing one prototype from each different class, and iteratively add instances having the highest margin of classification, or delete prototypes from the prototype set if they are not necessary for the correct classification of any training instance. This approach has accommodate outliers and has convergence characteristics that permit to achieve a minimum consistent set.

In [37], Gates proposed the Reduced Nearest Neighbor rule (RNN), which is a variant of the condensed methods. With RNN, the condensed set is obtained by iteratively contracting the initial training dataset. Each training instance is deleted from the condensed set if its deletion does not affect the correct classification of other instances. In [13], a novel approach is introduced to find prototypes for a nearest neighbor classifier. The idea is to start with every sample in the given training set as a prototype, and then merge nearest neighbors of the same class as long as the merge does not increase the error rate. The merging is accomplished by either simple or weighted averaging of the nearest neighbor pairs. This method decreases the size of the training set to a much greater extent than approaches considered so far [**?**].

In [90], Tomek proposed two modifications to the CNN approach in order to improve the selection of initial prototypes. The idea behind these modifications was to grow the condensed set using only instances that are close to the decision boundary. The subset obtained with these modifications is smaller than that obtained by the CNN algorithm, and the retained boundary instances are better chosen as they are closer to the decision boundary. In [21], Dasarathy proposed a method to find a Minimal Consistent Set (MCS) of instances based on the concept of the nearest unlike neighbor subset. This method uses a voting mechanism to select samples in the order of significance of their contribution to achieve the same performance of the 1-NN rule on the reduced set.

In [108], Wu et al. proposed an effective technique to speed up k-NN classification while maintaining the same level of accuracy. The Improved k-NN (IkNN) algorithm starts with an initial pattern set that is sufficiently large. It defines weights and attractive capacities (the alternative capacity of an instance $\mathbf{x}$ is defined as the number of patterns in the new set that matches the class of $\mathbf{x}$) for each pattern in the training set. IKNN is implemented by iteratively eliminating patterns which exhibit high attractive capacities. This approach tends to accelerate the classification procedure considerably, especially in cases where the feature space has large dimensionality.

In [32], Fayed and Atiya presented the Template Reduction for k-NN (TRKNN). TRKNN is based on the concept of nearest neighbors chains. For each pattern $\mathbf{x}$ in the training set, the chain C is built by finding the nearest neighbor of $\mathbf{x}$ from a different class. Then, the nearest neighbor of that new pattern belonging to the class of the starting instance $\mathbf{x}$ is located. The chain C is ended with two patterns that are nearest neighbor to each other. By calculating the distances between patterns inside the chain C, TRKNN drops the further patterns having the same class of $\mathbf{x}$. As the chain converges onto the boundary elements, TRKNN drops instances that are far away from the decision boundaries. This approach has the advantage of reducing the number of prototypes while maintaining the same level of classification accuracy as the traditional k-NN.

Hybrid methods

Hybrid methods aim at finding a minimal-size training subset that maintains or increases classification accuracy for test data, by removing noisy samples and superfluous instances.

In [1], the authors proposed a family of algorithms called Instance-Based learning, (IB1, IB2, IB3), which generates classification predictions by using specific instances. Using the classical nearest neighbor rule, the IBL algorithm starts with an initial empty set. Each instance in the training set is then added to it, based on its contribution to maximizing classification accuracy on subsequently presented instances. IB1 uses the 1-NN rule as a base classifier. IB2 adds to the reduced set only instances that are misclassified, to eliminate superfluous instances. However, IB2 is sensitive to noisy instances. These instances can be stored as they are exceptions and are generally misclassified. IB3 is an extension of IB2 that employs a significance test to distinguish noisy instances from instances that are good classifiers in the reduced set.

A class of techniques based on reducing storage requirements in instance-based learning algorithms was proposed in [106]. The *Decremental Reduction Optimization Procedure* (DROP) contains five methods (DROP1-DROP5) and introduces new heuristics to decide which instances to keep and which instances to remove from the training set. Each instance $\mathbf{x}$ has k-nearest neighbors and a nearest enemy which is the nearest instance to $\mathbf{x}$ with a different output class. Instances having $\mathbf{x}$ in their k-nearest neighbors are called associates of $\mathbf{x}$. The basic rule here is to remove $\mathbf{x}$ if at least as many of its associates would be correctly classified without $\mathbf{x}$. Whenever any instance $\mathbf{x}$ is removed, all of its associates must eliminate $\mathbf{x}$ from their list of nearest neighbors, and then must add a new nearest neighbor from the training set. The process has to be repeated for all training instances. Using the previous rule, the five algorithms (DROP1-DROP5) take careful note of the order in which instances are removed.

Genetic algorithms were introduced in [45][54] for simultaneous editing and feature selection to design the 1-NN classifier. The aim was to generate the minimal reduced set having the maximal classification accuracy. In [113], the authors proposed an approach to treat the reference subset selection as an optimization problem, which is to minimize sample size while keeping the resubstitution error rate below some threshold. They proposed to solve this problem using a Tabu Search (TS) algorithm [38]. When the error rate threshold is equal to zero, the algorithm outputs a near minimal consistent subset. While the threshold is set to a small appropriate value, the obtained reference subset may have reasonably good generalization capacity. A neighborhood exploration method is then introduced to improve the performance of the general TS.

In [102], the authors presented a method for selecting prototypes for the k-NN classifier using the Multi-category Proximal Support Vector Machine (MCPSVM). This approach tends to reduce the size of the training dataset in two ways: increases the separation between classes by eliminating noisy instances at the decision boundaries, and it removes instances that are far from boundaries and are often not useful for classification. In [56], the authors presented an alternative method of prototypes selection that merges the condensing and editing methods based on support vectors. This method selects prototypes only from support vectors. The support vectors found in the procedure are post processed with the DROP2 algorithm in order to obtain the resulting prototype set.

4.3 Editing multi-labelled data using the k-NN rule

4.3.1 Motivation

In multi-label learning, the goal is to generate a multi-label classifier that will generalize from a set of multi-labelled training instances in such a way that classification performances for labelling new data are optimized. However, errors in multi-labelled training datasets can occur for several reasons. One cause is the subjectivity, when the boundaries of each class are based on individual perspectives. For example, in genre classification of musical signals, each musical genre may have its boundaries shifted from person to person [5]. A second cause of anomalies or noisy instances is ambiguity during data-entry. For example, in clinical text for multi-label classification (medical multi-labelled data collected from Cincinnati children's hospital medical center), abbreviations and acronyms used to anonymization of patients may lead to ambiguity when processing such data by taking more than one sense and having multi-purposes (in a clinical setting, FT can be an abbreviation for *full-term*, *foot test*, *field test*, *full-time* or *family therapy*) [64]. Other errors can arise from missing information and data transformation or storage. Furthermore, many examples may have an erroneous set of labels due to an experimental assignment problem or even a human annotation error. To the best of our knowledge, no algorithm addressing these problems under the multi-label framework has been proposed so far.

In the following, we propose an original method to edit multi-labelled data by identifying and eliminating erroneous or anomalous samples. The purpose of this method is three-fold: first, to increase the quality of training instances assumed to become more reliable; second, to improve the performances of the classifier built from the resulting training data; and third to increase the response time of the learning algorithm. This method is based on the k-nearest neighbor rule for multi-label classification, and on an evaluation criterion used locally in the set $\mathcal{N}_{\mathbf{x}}$ of k-nearest neighbors of $\mathbf{x}$ to evaluate the quality of an instance $\mathbf{x}$. Based on this evaluation criterion, we can delete the most irrelevant, or the *worst* samples from the initial training dataset. We introduce in the next section the Hamming loss which is an evaluation criterion that we selected to evaluate relevancy of samples.

4.3.2 Hamming loss

As explained in Section 1.6, in traditional single-label classification the predictive performance is usually measured by the test error rate; while in multi-label classification the predictive performance is more complex, because the classification of each test instance can be fully correct, partially correct or fully wrong. Given a set $\mathcal{S} = \{(\mathbf{x}_1, Y_1), \ldots, (\mathbf{x}_m, Y_m)\}$ of m test examples, evaluation metrics can be divided into two groups: *prediction-based* and *ranking-based* metrics [94]. We will focus in this section on the Hamming loss metric. The Hamming loss is a prediction-based metric regarded as an average of the error rate of the classifier on the Q binary problems where the decision is performed separately [76]. It is defined by:

$$H = \frac{1}{m} \sum_{i=1}^{m} \frac{|Y_i \triangle \widehat{Y}_i|}{Q}, \tag{4.1}$$

where Y_i is the ground truth label set for the pattern $\mathbf{x}_i$, $\widehat{Y}_i$ is the predicted label set for $\mathbf{x}_i$ and $\triangle$ denotes the symmetric difference between two sets. In other words, the Hamming loss is based on counting prediction errors (an incorrect label is predicted) and missing errors (a true label is not predicted). Note that the value of this criterion is in the interval [0, 1] and smaller values correspond to higher classification quality. We will present hereafter a simple method using this metric conjointly with a k-NN rule for multi-label classification in order to edit the training dataset.

4.3.3 Editing algorithm: Edited Nearest Neighbor for Multi-Labelled data

Let $\mathbf{x}$ be an unseen instance for which we wish to estimate the set of labels. Given a training set $\mathcal{D} = \{(\mathbf{x}_1, Y_1), \ldots, (\mathbf{x}_n, Y_n)\}$ where $Y_i \subseteq \mathcal{Y} = \{\omega_1, \ldots, \omega_Q\}$. In the following steps, we describe the proposed method to edit the training dataset:

1. For each training instance $\mathbf{x}_i$ in $\mathcal{D}$, search for $\mathcal{N}_{\mathbf{x}_i}$, the set of its k nearest neighbors according to some distance function (e.g. the euclidian distance);
2. Apply a k-NN based multi-label classifier and calculate a predicted set of labels $\widehat{Y}_i$ for $\mathbf{x}_i$;
3. For each training instance in $\mathcal{D}$, calculate the associated Hamming loss given by:
$$\mathcal{H}Loss_i = \frac{|Y_i \triangle \widehat{Y}_i|}{Q}; \tag{4.2}$$
4. Estimate the Hamming loss $HLoss$, which is the mean of the associated Hamming loss for all instances in $\mathcal{D}$:
   – if $HLoss$ is less than a predefined threshold t, then stop the algorithm;
   – else,
   (a) Rank the training instances in $\mathcal{D}$ with respect to their $HLoss_i$ and select a subset $\mathcal{E}^l$ containing l instances with the higher Hamming loss $HLoss_i$;
   (b) Update the training set by deleting those in $\mathcal{E}^l$: $\mathcal{D} \leftarrow \mathcal{D} \setminus \mathcal{E}^l$;
   (c) Return to step 1.

Note that any k-NN based multi-label classifier [26][111][115] can be applied in Step 2. In this Chapter, we chose the EMLkNN method introduced in Chapter 2. According to this method, each element in $\mathcal{N}_{\mathbf{x}_i}$ represents a piece of knowledge about the labelling of $\mathbf{x}_i$. A two-valued mass function is then associated to each of the k neighbors in $\mathcal{N}_{\mathbf{x}_i}$ according to Equations (2.28) and (2.29):

$$m_i(A_i, B_i) = \alpha \exp\left(-\gamma d(\mathbf{x}, \mathbf{x}_i)\right), \tag{4.3}$$
$$m_i(\emptyset, \emptyset) = 1 - \alpha \exp\left(-\gamma d(\mathbf{x}, \mathbf{x}_i)\right), \tag{4.4}$$

where A_i is the set of labels that surely apply to instance $\mathbf{x}_i$, and B_i is the set of labels that surely do not apply to the same instance (A_i, $B_i \subseteq \Omega$). These items of evidence are combined to produce a global mass function, using the conjunctive rule of Equation 2.30:

$$m = \bigcirc\!\!\!\!\cap_{i:\mathbf{x}_i \in \mathcal{N}_{\mathbf{x}}^k} m_i.$$

In order to estimate the label set for $\mathbf{x}_i$ denoted by $\widehat{Y}_i$, the global mass function is used according to Equation 1.31:

$$\widehat{Y}_i = \{\omega \in \mathcal{Y} \mid bel(\{\omega\}, \emptyset) \geq bel(\emptyset, \{\omega\})\}.$$

Intuitively, k should be set to a small value because if k is high, undesirable instances elimination will occur on the boundary between different classes. If k is equal to 1, the EMLkNN algorithm boils down to the 1-NN algorithm, and the set of labels to be assigned to an example is the same as that of his neighbor. In Step 3, one can use other stopping criteria than the general $HLoss$. For example, we can stop editing if the Hamming loss associated to each instance is less than a predefined threshold t. We can also substitute the Hamming loss by another multi-label metric evaluation. In Steps 4a and 4b, we delete instances with high value of $HLoss_i$, which means deleting the *worst* instances with respect to a local EMLkNN rule. One can add a condition to keep instances belonging to classes with low occurrence.

4.3.4 Effect of editing on Rank-SVM and EMLkNN

To study the efficiency of our editing method, we analyzed the effect of applying multi-label classification methods on the reduced set obtained by editing initial training datasets. We chose two well-known algorithms from the literature of multi-label learning. The first one, RankSVM, is based on the concept of multi-label ranking, while the other one, EMLkNN, presented in Chapter 2, is based on the evidential multi-label k nearest neighbors concept.

RankSVM is a multi-label ranking approach introduced by Elisseeff and Weston in [31]. The ultimate goal was to minimize a criterion measure for multi-label learning, called Ranking loss, and to maximize the margin. The Ranking loss is defined by:

$$\mathcal{R}Loss = \frac{1}{m}\sum_{i=1}^{m} \frac{1}{|Y_i||\overline{Y_i}|}|R(\mathbf{x}_i)|, \tag{4.5}$$

where $R(\mathbf{x}_i) = \{(\omega_q, \omega_l) \in Y_i \times \overline{Y_i} \mid f(\mathbf{x}_i, \omega_q) \leq f(\mathbf{x}_i, \omega_l)\}$, $\overline{Y_i}$ denotes the complement of Y_i in $\mathcal{Y}$, and f is the scoring function that gives a score for each label ω_q interpreted as the probability that ω_q is relevant. The authors introduce a special multi-label margin defined on $(\mathbf{x}, Y)$ as the signed distance between the instance $\mathbf{x}$ and the decision boundary defined by: $\{\mathbf{x} \mid \exists q, \langle w_q - w_l, \mathbf{x}\rangle + b_q - b_l = 0\}$. It is equal to:

$$\min_{(q,l) \mid (\omega_q,\omega_l)\in(Y_i\times\overline{Y_i})} y_q \frac{\langle w_q - w_l, \mathbf{x}\rangle + b_q - b_l}{\|w_q - w_l\|}$$

where w_q, w_l and b_q, b_l denote the weight vectors and bias terms, and y_q is a binary element equal to $+1$ if label q is in Y, -1 otherwise.

The Rank-SVM model is built from two different sub-systems. The first one, named ranking system, is constructed as follows:

$$\min_{w_j, j=1,\ldots,Q} \sum_{q=1}^{Q} \|w_q\|^2 + C \sum_{i=1}^{n} \frac{1}{|Y_i|\,|\overline{Y_i}|} \sum_{(q,l)\ \mid\ (\omega_q,\omega_l)\in(Y_i\times\overline{Y_i})} \xi_{iql} \tag{4.6}$$

subject to:

$$\begin{aligned} \langle w_q - w_l, \mathbf{x}_i\rangle + b_q - b_l &\geq 1 - \xi_{iql},\ (q,l) \in Y_i \times \overline{Y_i} \\ \xi_{iql} &\geq 0, i = 1,\ \ldots, n, \end{aligned}$$

where the penalty constant C controls the trade-off between the slack variable ξ_{iql} and the margin. Using the Lagrangien technique, the dual of (4.6) is formulated as:

$$\max_{\alpha_{iql}} W(\alpha) = -\frac{1}{2}\sum_{q=1}^{Q}\sum_{h,i=1}^{n} \beta_{qh}\beta_{qi} < \mathbf{x}_h, \mathbf{x}_i > + \sum_{i=1}^{n} \sum_{(q,l)\ \mid\ (\omega_q,\omega_l)\in(Y_i\times\overline{Y_i})} \alpha_{iql}$$

subject to:

$$\begin{gathered} \alpha_{iql} \in \left[\frac{C}{C_i}\right] \\ \textstyle\sum_{i=1}^{n} \sum_{(j,l)\ \mid\ (\omega_j,\omega_l)\in(Y_i\times\overline{Y_i})} c_{ijl}\alpha_{ijl} = 0,\ q = 1, \ldots, Q, \end{gathered}$$

with

$$c_{ijl} = \begin{cases} 0 & \text{if } j \neq q \text{ and } l \neq q \\ +1 & \text{if } j = q \\ -1 & \text{if } l = q \end{cases}$$

and

$$\beta_{qi} = \sum_{(j,l)\ \mid\ (\omega_j,\omega_l)\in(Y_i\times\overline{Y_i})} c_{ijl}\alpha_{ijl}.$$

The ranking system orders the labels according to their outputs, $r_q(x) = \langle w_q, \mathbf{x}\rangle + b_q$ for $q = 1, \ldots, Q$. The other goal of the Rank-SVM method is to predict a threshold $t(\mathbf{x})$ and all integer q such that $r_q(\mathbf{x}) > t(\mathbf{x})$ are considered to belong to the label set Y of $\mathbf{x}$. It is well-known that such an algorithm can be generalized to non-linear separating boundaries by just replacing the dot products $< \mathbf{x}_i, \mathbf{x}_j >$ by kernels $k(\mathbf{x}_i, \mathbf{x}_j)$.

Rank-SVM on edited dataset

The Rank-SVM method described above is based on the margin criterion. SVM-based classifiers separate different classes by finding optimal classification hyperplanes from the training dataset. If the dataset contains misclassified instances, the SVM-based algorithm may lead to erroneous separating boundary (hyperplane in the linear case). The problem can be addressed by using a soft margin that accepts some misclassification instances from the training dataset. This can be achieved by introducing positive slack variables (ξ) as in Equation (4.6), where C is the tradeoff between the model complexity (having a large margin) and the empirical error (having smaller errors in the training dataset). If the parameter C is high, this method will try to correctly classify all training examples to the detriment of generalization performance. When C is decreased, instances near the boundaries become margin errors, providing a much larger margin for the rest of the data. Since the edited training dataset is expected to contain fewer noisy data than the original dataset, it is straightforward to see that learning a Rank-SVM to multi-label classification from the edited dataset will be more simple. In fact, the number of slack variables (ξ) will be reduced and smoother decision boundaries with better generalization ability can be expected.

EMLkNN on edited dataset

The EMLkNN method can be applied using the edited training dataset in order to classify unseen instances. The number k of neighbors to be used is not necessarily the same as that used in the editing algorithm. To avoid confusion, the number of neighbors used in the editing algorithm will be noted by k' if needed. It is easy to see that by using the edited training dataset, the number of neighbors to be used by the EMLkNN method can be less than that used by the same method on the original training dataset. This will decrease the running time of the algorithm since less distances and masses combination have to be calculated. More generally, the objective of using an edited training dataset with the EMLkNN method is three-fold: improving the performance of multi-label classifiers, using less memory for storing training instances and distances, and providing faster decision rule.

Table 4.1: Description of the synthetic data without the erroneous instances.

Label set	Training instances	Testing instances
$\{\omega_1\}$	85	41
$\{\omega_2\}$	84	41
$\{\omega_3\}$	82	46
$\{\omega_1, \omega_2\}$	30	20
$\{\omega_1, \omega_3\}$	42	18
$\{\omega_2, \omega_3\}$	46	23
$\{\omega_1, \omega_2, \omega_3\}$	31	11

4.4 Experimental Evaluation

In this section, we present experiment results with synthetic and real-world datasets from different domains to demonstrate the effect of edition on the performances of the two multi-label classification methods described below.

4.4.1 Experiments with Synthetic Data

In this section, we will illustrate the behavior of our editing algorithm on synthetic datasets using the two methods of classification discussed above. The goal of these experiments is to study the effects of edition on multi-label learning algorithms.

A dataset with three-overlapping classes in two-dimension was first considered. The dataset contains 600 instances belonging to three possible labels $\Omega = \{\omega_1, \omega_2, \omega_3\}$. These instances were drawn from seven Gaussian distributions with means $(-5, -5)$, $(5, -5)$, $(0, 5)$, $(0, -5)$, $(-3, 1)$,$(3, 1)$, and $(0, 0)$. The standard deviations was equal two for the first three distributions and one for the others. We assigned the following classes, respectively, for samples drawn from each of these distributions: $\{\omega_1\}$, $\{\omega_2\}$, $\{\omega_3\}$, $\{\omega_1, \omega_2\}$, $\{\omega_1, \omega_3\}$, $\{\omega_2, \omega_3\}$, $\{\omega_1, \omega_2, \omega_3\}$. This dataset was randomly divided into training and test datasets with size 400 and 200, respectively. Table 4.1 gives the distribution of instances over the different labels.

To test our editing algorithm, 40 instances drawn in the region allocated to classes $\{\omega_1\}, \{\omega_2\}$ and $\{\omega_1, \omega_2\}$ were wrongly assigned to class $\{\omega_3\}$. These noisy samples are generated randomly from two normal distributions with means $(-4, -6)$ and $(4, -6)$, respectively, and a standard deviation equal to 2. Figure 4.2 shows the dataset (initial + noisy instances) with their class assignments.

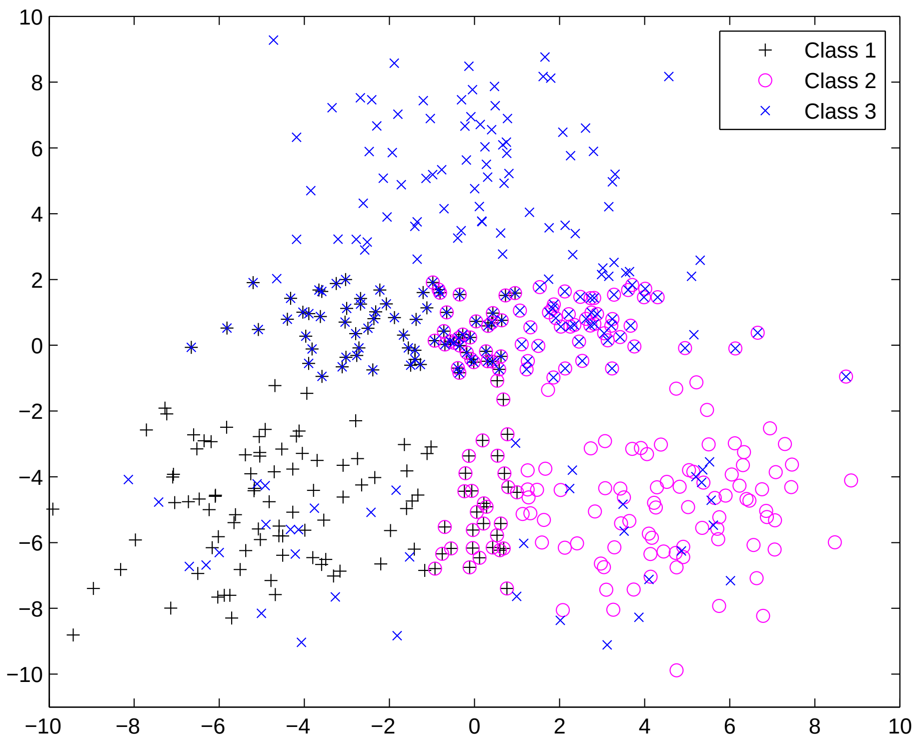


Figure 4.2: Training instances of synthetic data

Figures 4.3 and 4.4 show the decision boundaries for our synthetic data with a support vector domain using a Gaussian kernel. The boundary region for each class label was drawn using the Rank-SVM method, with the Gaussian kernel: $k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma_r \|\mathbf{x} - \mathbf{x}'\|^2)$, $\gamma_r = 5$. We used the same parameters values for the Rank-SVM method with the training data before and after editing. Figure 4.3 shows the decision boundaries for the initial training dataset. As we can see, these decision boundaries are significantly influenced by noisy instances and there is no clear separation between classes. In the area of class $\{\omega_2\}$ (on the right of this figure), we can see several zones belonging to classes $\{\omega_1, \omega_2, \omega_3\}$. Also, in the area of class $\{\omega_1\}$, the erroneous instances create three zones with instances belonging to class $\{\omega_3\}$.

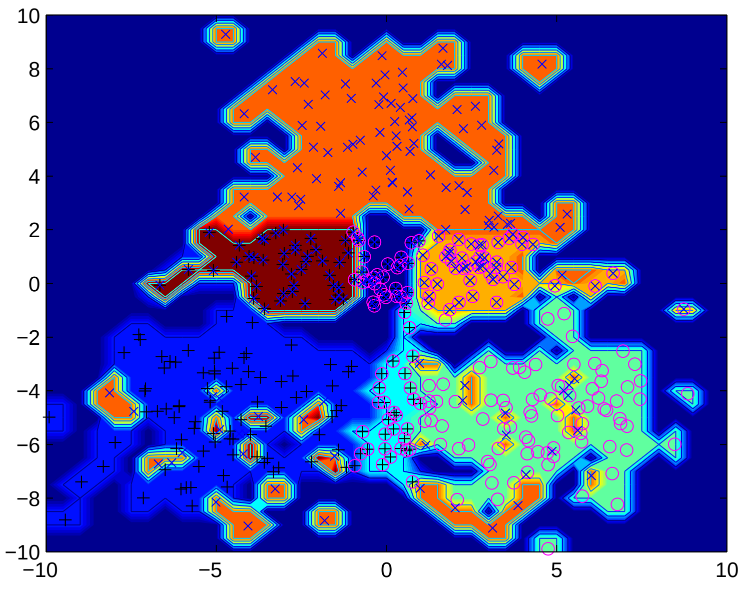


Figure 4.3: The Rank-SVM decision boundaries between classes with the training instances before editing (Gaussian kernel).

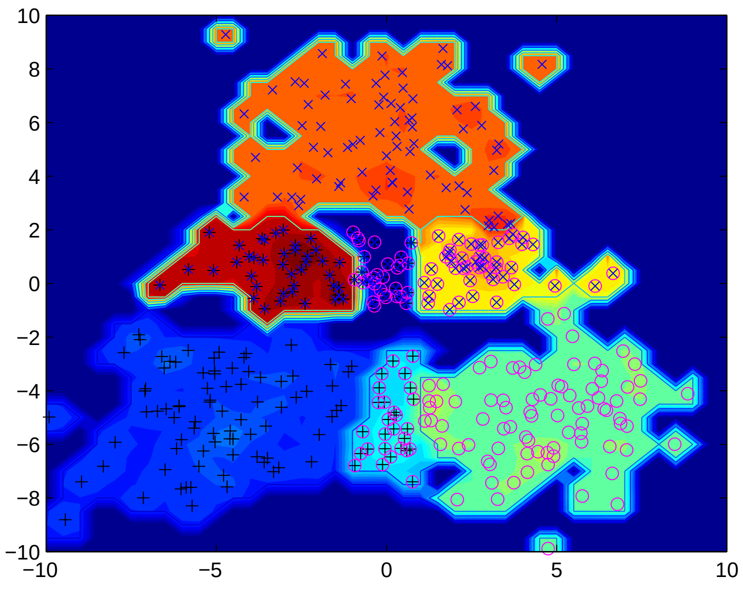


Figure 4.4: The Rank-SVM decision boundaries between classes with the training instances after editing (Gaussian kernel).

Table 4.2: Some evaluation measures for the Rank-SVM method before and after the edition of the synthetic dataset.

	Before Editing	After Editing
Hamming loss^{-}	0.1667	**0.1017**
Accuracy^{+}	0.7283	**0.8358**
F1^{+}	0.7422	**0.8423**

+(-): the higher (smaller) the value, the better the performance.

In Figure 4.4, we can see the decision boundaries for the same dataset after editing. In this figure, we can see that, with editing, noisy instances have been removed, and smoother decision boundaries are produced. The area is now divided into seven zones. Instances belonging to class $\{\omega_1\}$ are on the left of Figure 4.4, instances assigned by label $\{\omega_2\}$ are on the right, and instances labelled with $\{\omega_3\}$ are at the top. Using a geometrical interpretation, we can easily distinguish the area belonging to each combination of these classes. Instances annotated by three classes are in the middle of this figure. Note that the number of training instances was reduced to 382 (initial number of training data was 440), and the number of support vectors was decreased from 409 to 352. Table 4.2 reports the experimental results on three evaluation criteria: Hamming loss, accuracy and the F1-measure.

From these two figures, we can observe that training the Rank-SVM method with a purified dataset leads to smoother separating boundaries, creates homogeneous clusters and reduces the number of support vectors.

Figure 4.5 shows the performance of our editing approach on the synthetic data using the EMLkNN method. We used from the library of multi-label measures three evaluation criteria: Hamming loss, accuracy and the F1-measure. The values of these metrics are shown as a function of the number of neighbors k. From this figure, we can observe that when k takes small values, the EMLkNN algorithm tested on the edited dataset performs better than EMLkNN tested on noisy dataset. As k increases, the EMLkNN method tends to have the same performance on these two datasets. This can be explained by the fact that, when increasing the number of neighbors, the effect of randomly erroneous instances decreases giving that we use more information (coming from more instances), and also the applied method (EMLkNN) is based on an evidential distance-weighted k-nearest neighbor rule.

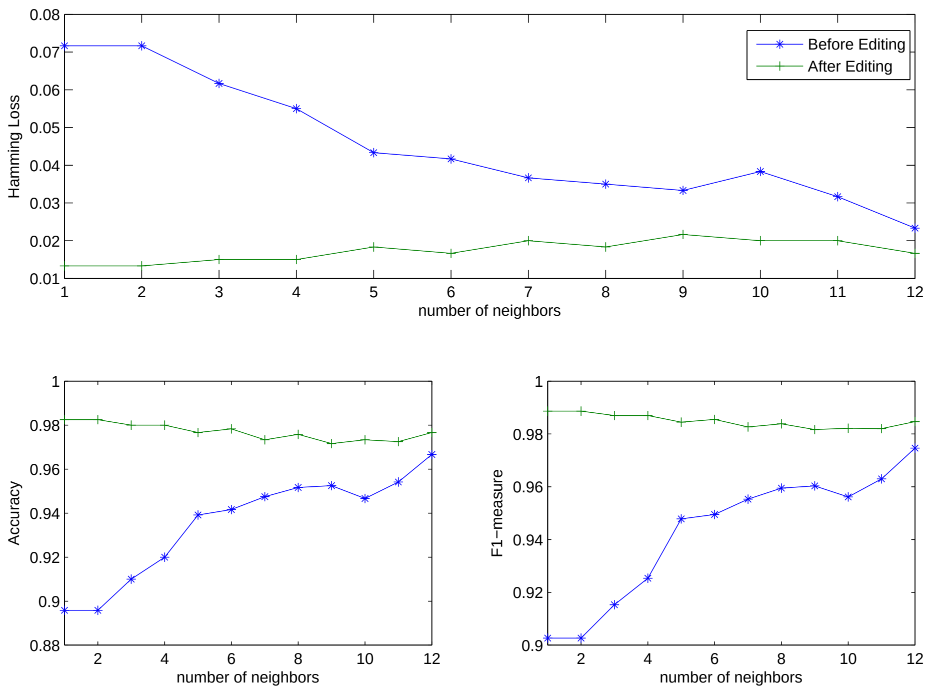


Figure 4.5: Some evaluation measures for the EMLkNN method before and after the edition of the synthetic dataset.

4.4.2 Experiments on Real-World Data

In this section, we apply the two multi-label classification methods discussed above (EMLkNN and Rank-SVM) to our datasets and we evaluate their performances before and after editing. In the following, we will report the benchmark datasets, the evaluation metrics used in our experiments and parameter settings for edition. Finally, we will provide a discussion of experimental results.

Datasets

The datasets[1] that were included in our experiments cover different application domains: multimedia classification (Emotions), bioinformatics (Yeast) and text cate-

1. Datasets available at `http://mulan.sourceforge.net/datasets.html`, and `http://cse.seu.edu.cn/people/zhangml/`.

gorization (Medical, Enron and Webpage).

- *Emotions dataset.* This dataset contains 593 songs described by eight rhythmic features and 64 timbre features. There are six classes, and each song can belong to more than one label according to the Emotions generated [91]. More details were given in Chapter 3.
- *Yeast dataset.* The *yeast Saccharomyces cerevisiae* is one of the best studied organisms. Each gene is described by the concatenation of micro-array expression data and phylogenetic profile and it is associated with a subset of 14 functional classes from the Comprehensive Yeast Genome Database of the Munich Information Center for Protein Sequences [1]. This dataset contains 2417 genes and 14 possible labels [63].
- *Medical dataset.* This dataset contains 978 documents for patient symptom histories collected from the *Computational Medicine Center* concerning a challenge task on the automated processing of clinical free text. Each document is represented by a vector of 1449 features [64].
- *Enron dataset.* The *Enron email*[2] dataset was made public by the Federal Energy Regulatory Commission during its investigation. It contains around 517.431 emails (without attachments) from 151 users distributed in 3500 folders. Each message includes the senders and the receiver email address, date and time, subject, body, text and some other email specific technical details. After preprocessing and careful selection of these documents, 53 different labels are obtained with 753 combinations of distinct label sets [82].
- *Webpage categorization dataset.* This dataset were collected from the "yahoo.com" domain [88]. Eleven different webpage categorization subproblems are considered, corresponding to 11 independent multi-label categories: *Arts and Humanities, Business and Economy, Computers and Internet, Education, Entertainment, Health, Recreation and Sports, Reference, Science, Social and Science, and Society and Culture.* Each subproblem consists of 5000 documents (2000 as training dataset and 3000 as testing dataset). Each webpage was represented as a bag of words and normalized to the unit length.

1. `http://mips.gsf.de/genre/proj/yeast/`
2. `http://enrondata.org/content/research/`

Table 4.3: Characteristics of the Emotions, Yeast, Medical and Enron datasets.

	Domain	Number of instances	Feature vector dimension	Number of labels	Label cardinality	Label density	Distinct label sets
Emotions	music	593	72	6	1.868	0.311	27
Yeast	biology	2417	103	14	4.237	0.303	198
Medical	text	978	1449	45	1.245	0.028	94
Enron	text	1702	1001	53	3.378	0.064	753

Table 4.4: Characteristics of the Webpage categorization dataset.

	Number of instances	Feature vector dimension	Number of labels	Label cardinality	Label density	Distinct label sets
Arts and Humanities	5000	462	26	1.627	0.063	462
Business and Economy	5000	438	30	1.590	0.053	161
Computers and Internet	5000	681	33	1.487	0.046	253
Education	5000	550	33	1.465	0.044	308
Entertainment	5000	640	21	1.426	0.068	232
Health	5000	612	32	1.667	0.052	257
Recreation and Sports	5000	606	22	1.414	0.065	322
Reference	5000	793	33	1.159	0.035	217
Science	5000	743	40	1.489	0.036	398
Social and Science	5000	1047	39	1.274	0.033	226
Society and Culture	5000	636	27	1.705	0.063	582

Tables 4.3 and 4.4 provide an overview of the different characteristics of all experimental datasets.

Evaluation measures

To evaluate the performance of our proposed editing algorithm with the two multi-labelled methods discussed above, several measures are employed. These measures can be categorized into two groups: prediction-based and ranking-based metrics (more details are given in Chapter 1). In this chapter, we consider the following measures:

– Prediction-based: Hamming loss, Accuracy, Precision, Recall and F1-measure;

– Ranking-based: Average precision, Coverage, Ranking loss, and One-error.

Parameter Tuning

In this section, we comment how to tune different parameters to apply the different algorithms described in this paper. Note that we call *editing* parameters those applied on the initial dataset with the editing algorithm in order to obtain an edited training

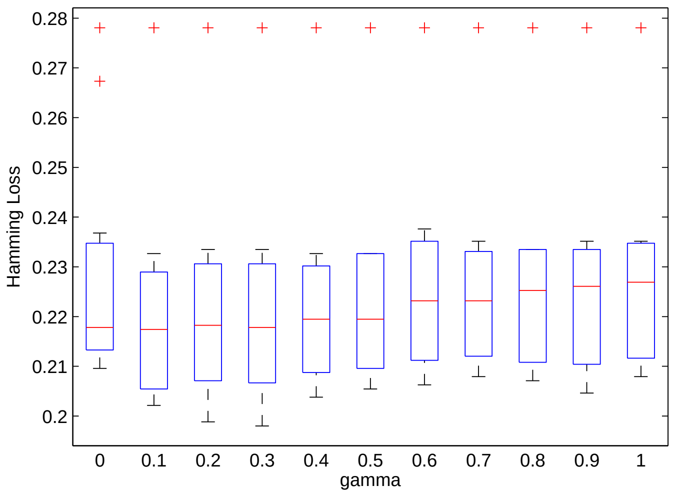


Figure 4.6: Hamming loss measure for EMLkNN on the initial Emotions training set for different values of γ.

dataset. We call *testing* parameters those used in a multi-label classification algorithm learnt from initial or edited learning dataset. Hereafter, we will show the influence of these parameters by using the *Emotions* dataset.

Editing parameters

For the editing algorithm presented in Section 4.3.3, there are three tunable parameters:

- γ: Parameter used in Equations (4.3) and (4.4) to scale the distance to each neighbor. It was fixed at the best value obtained by cross validation using the EMLkNN method on the initial training dataset.
- k': Number of neighbors used in the editing algorithm.
- t: Threshold used to determine the number l of instances to delete. We use in the simulation a Hamming loss calculated on each instance as in Equation (4.2). This Hamming loss calculated on only one instance will have a value equals q/Q, where $q \in \{0, \dots, Q\}$. Note that the value of the parameter t to be taken should depend on the global Hamming loss calculated on the training dataset.

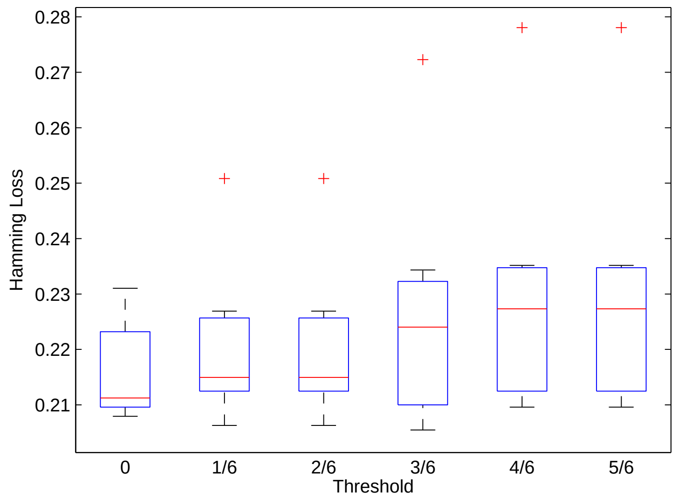


Figure 4.7: Hamming loss measure for EMLkNN on the initial Emotions training set as a function of t.

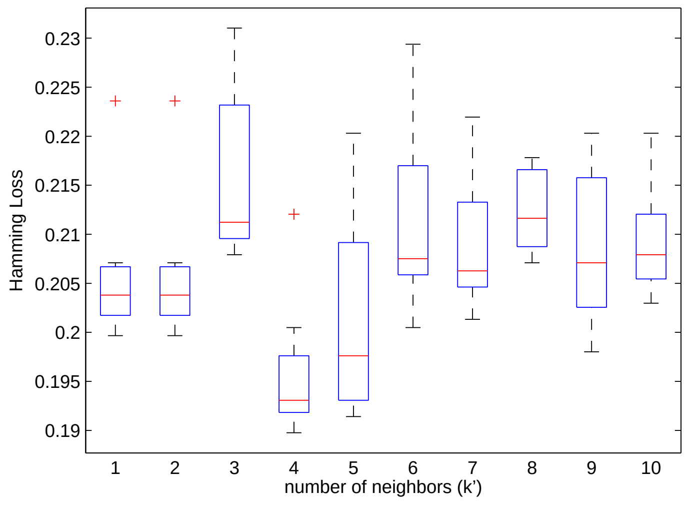


Figure 4.8: Hamming loss measure for EMLkNN on the initial Emotions training set as a function of k'.

Figure 4.6 shows the box plot for the Hamming loss metric obtained by the EMLkNN method on the initial dataset before editing the data for different values of γ, where k was varied from 1 to 12 (thus each box plot corresponds to 12 values of the Hamming loss obtained for a given γ). Figure 4.7 shows the Hamming loss measure obtained as a function of t, where k' was varied from 1 to 12, γ was fixed to 0.1, and k was fixed to 3 (we can get the same results for any value of k, for that we chose for it a small value). The box plot in Figure 4.8 shows the Hamming loss criterion with respect to the number of neighbors k'. k was varied from 1 to 12, and γ was fixed to 0.1.

Testing parameters

In the testing phase, the EMLkNN and the Rank-SVM methods are tested with the edited data. EMLkNN has two parameters: the number of neighbors k, and the discounting parameter γ. These parameters were determined using grid search and by focusing on the Hamming loss measure: k was varied from 1 to 12, and γ from 0 to 1 with 0.01 steps. Note that the algorithm presented in Section 4.3.3 was repeated only once by taking a small value of t to eliminate an important number of erroneous instances at once.

For the Rank-SVM method, we used the Gaussian kernel with three tunable parameters: kernel scale parameter γ_r, penalty constant C, and maximal iterations M. By focusing on the Hamming loss measure, cross-validation via grid search was applied for parameter tuning as explained in [110]. The γ_R and C parameters took values from $2^7, 2^6, \ldots,$ to 2^{-7} respectively. M was set to 50, 100, 150 and 200.

Results and Discussion

In this section, we evaluate the performance of the editing algorithm by comparing the results achieved by the EMLkNN and Rank-SVM methods before and after editing. Using the optimal parameter values obtained on the training datasets, we studied the performance using independent test datasets. The experimental results on datasets are given in Tables 4.5-4.8. For the webpage dataset, the average performance out of the 11 different categorization problems is reported in Table 4.9. The rank of each method is given, and the best value on each evaluation criterion is highlighted in bold letters. These results can be summarized as follows:

– From the results in Tables 4.5-4.9, we can observe that EMLkNN applied on edited data improves the performance of the same method on the initial dataset for all prediction-based metrics except the Hamming loss measure. The Hamming loss criterion is similar before and after edition.
– Regarding the Rank-SVM method, results on editing datasets are better than those on initial datasets for all measures (prediction-based and ranking-based metrics).
– The Rank-SVM applied on editing datasets gives the best performance according to the majority of evaluation measures for the Yeast, Medical, Enron, and Web-page datasets. For the Emotions dataset, the best performance on the ranking-based measures was obtained by the Rank-SVM method applied to the edited dataset, while the best results according to predicted-based measures were obtained by the EMLkNN algorithm applied to the edited dataset.

In order to show the effect of edition on data storage, Table 4.10 reports the number of instances of full and edited training datasets. The results indicate that the edited datasets require less storage space than do the initial datasets.

In order to study the impact of edition on classification time, we compared the time used by each method (programmed in Matlab) applied to the initial and edited datasets. Table 4.11 presents the total running time (learning + testing time) using the initial training and edited datasets. We can see that the running time of the two classifiers (EMLkNN and Rank-SVM) are significantly reduced in our experiments, except for the Enron dataset. In general, EMLkNN is faster than Rank-SVM, due to the space complexity of the Rank-SVM method which is proportional to $n * Q^2$. The machine used was Intel(R) Xeon(R) CPU at 2.67 GHz, 12 GB RAM with Matlab2012a.

To statistically measure the significance of performance difference between results on initial datasets and those on edited datasets, pairwise t-tests at 5% significance level are carried out using ten-fold cross validation. The average results are reported in Tables 4.12-4.15.

The results presented in this section show the advantage of editing multi-label datasets to improve the performance of multi-label classifiers. By comparing the performance of multi-label classifiers (EMLkNN and Rank-SVM) before and after edition, we can conclude that editing initial multi-label datasets improve the performance evaluation of some classifiers. Furthermore, we may reduce the complexity of classifiers since we need to train less instances, which are distributed into more homogeneous clusters. We might deduce also that editing training datasets is a way to reduce the running time complexity of some multi-label classification methods. Even if we use the Hamming loss criterion to edit the training datasets, we can get better performance on other metrics.

4.5 Conclusion

In this chapter, we have addressed the problem of prototype selection in the framework of multi-label learning. Although the extensive works in multi-label classification, to the best of our knowledge, the topic of prototype selection has not received any attention so far. The goal is not only to optimize performance of some classifiers, but also the size of the training dataset must be reduced as well as the computational time of learning algorithms. This chapter demonstrates our contribution to edit multi-labelled dataset.

Edited Nearest Neighbor for Multi-Labelled data is an efficient editing method. The idea is, first, to classify all training instances using a k-NN rule, and, second, to eliminate erroneous instances based on a local criterion induced from the Hamming loss measure. The reduced set of instances is then used to classify unseen instances. We have demonstrated the effect of editing dataset on two learning algorithms: the EMLkNN and the Rank-SVM. This was illustrated through an example on synthetic data.

We applied our algorithm of editing to five real-world datasets from different domains of application: multimedia classification, bioinformatics and text categorization. Experimenting with these datasets, we observed that the learning algorithms (EMLkNN and Rank-SVM) with the editing datasets significantly outperformed the same algorithms on the initial datasets in terms of classification performance and computational costs. The explanation is that the editing datasets are distributed in more homogeneous clusters by reducing the number of irrelevant instances. Learning from these new instances is faster with better generalization ability.

Table 4.5: Experimental results on the Emotions dataset.

	EMLkNN		Rank-SVM	
	Before Editing	After Editing	Before Editing	After Editing
Hamming loss$^-$	0.209(4)	0.204(2)	0.206(3)	**0.194**(1)
One-Error$^-$	0.287(2)	0.297(3)	0.302(4)	**0.277**(1)
Coverage$^-$	1.881(2)	2.010(4)	1.896(3)	**1.847**(1)
Ranking loss$^-$	0.168(3)	0.220(4)	0.166(2)	**0.158**(1)
Average Precision$^+$	0.7994(2)	0.7959(4)	0.7993(3)	**0.8080**(1)
Accuracy$^+$	0.519(4)	**0.569**(1)	0.546(3)	0.561(2)
Precision$^+$	0.656(3)	**0.705**(1)	0.651(4)	0.690(2)
Recall$^+$	0.592(3)	**0.657**(1)	0.642(2)	0.642(2)
F1$^+$	0.596(4)	**0.648**(1)	0.621(3)	0.637(2)

+(-): the higher (smaller) the value, the better the performance.

Table 4.6: Experimental results on the Yeast dataset.

	EMLkNN		Rank-SVM	
	Before Editing	After Editing	Before Editing	After Editing
Hamming loss$^-$	0.205(4)	0.202(3)	0.197(2)	**0.193**(1)
One-Error$^-$	0.261(4)	0.249(3)	**0.221**(1)	0.238(2)
Coverage$^-$	6.494(3)	6.577(4)	6.424(2)	**6.269**(1)
Ranking loss$^-$	0.188(3)	0.201(4)	0.167(2)	**0.165**(1)
Average precision$^+$	0.751(3)	0.751(4)	0.767(2)	**0.768**(1)
Accuracy$^+$	0.515(4)	0.529(2)	0.522(3)	**0.539**(1)
Precision$^+$	0.685(4)	0.689(3)	0.697(2)	**0.703**(1)
Recall$^+$	0.599(4)	0.618(3)	0.625(3)	**0.635**(1)
F1$^+$	0.613(4)	0.627(3)	0.628(3)	**0.641**(1)

+(-): the higher (smaller) the value, the better the performance.

Table 4.7: Performance of our method on the Medical dataset.

	EMLkNN		Rank-SVM	
	Before Editing	After Editing	Before Editing	After Editing
Hamming loss$^-$	0.018(3)	0.018(4)	0.012(2)	**0.011**(1)
One-Error$^-$	0.285(2)	0.291(3)	**0.141**(1)	**0.141**(1)
Coverage$^-$	3.541(3)	3.661(4)	**1.135**(1)	1.255(2)
Ranking loss$^-$	0.124(3)	0.126(4)	**0.015**(1)	0.018(2)
Average precision$^+$	0.779(3)	0.776(4)	0.897(2)	**0.898**(1)
Accuracy$^+$	0.559(4)	0.585(3)	0.688(2)	**0.726**(1)
Precision$^+$	0.617(4)	0.647(3)	0.744(2)	**0.781**(1)
Recall$^+$	0.569(4)	0.594(3)	0.718(2)	**0.754**(1)
F1$^+$	0.581(4)	0.608(3)	0.716(2)	**0.754**(1)

+(-): the higher (smaller) the value, the better the performance.

Table 4.8: Performance of our method on the Enron dataset.

	EMLkNN		Rank-SVM	
	Before Editing	After Editing	Before Editing	After Editing
Hamming loss$^-$	0.057(3)	0.059(4)	0.055(2)	**0.053**(1)
One-Error$^-$	0.437(3)	0.478(4)	0.287(2)	**0.275**(1)
Coverage$^-$	21.959(3)	26.226(4)	13.758(2)	**12.772**(1)
Ranking loss$^-$	0.261(3)	0.395(4)	0.099(2)	**0.090**(1)
Average precision$^+$	0.568(3)	0.509(4)	0.619(2)	**0.647**(1)
Accuracy$^+$	0.303(4)	0.318(3)	0.398(2)	**0.436**(1)
Precision$^+$	0.473(4)	0.484(3)	0.574(2)	**0.587**(1)
Recall$^+$	0.340(4)	0.359(3)	0.495(2)	**0.556**(1)
F1$^+$	0.372(4)	0.390(3)	0.511(2)	**0.550**(1)

+(-): the higher (smaller) the value, the better the performance.

Table 4.9: Performance of our method on the Webpage dataset.

	EMLkNN		Rank-SVM	
	Before Editing	After Editing	Before Editing	After Editing
Hamming loss$^-$	0.054(3)	0.056(4)	0.043(2)	**0.042**(1)
One-Error$^-$	0.559(4)	0.551(3)	0.417(2)	**0.402**(1)
Coverage$^-$	11.036(3)	12.379(4)	5.080(2)	**4.169**(1)
Ranking loss$^-$	0.446(3)	0.520(4)	0.128(2)	**0.1000**(1)
Average precision$^+$	0.498(3)	0.482(4)	0.651(2)	**0.673**(1)
Accuracy$^+$	0.337(4)	0.363(3)	0.402(2)	**0.437**(1)
Precision$^+$	0.393(4)	0.423(3)	0.465(2)	**0.508**(1)
Recall$^+$	0.365(4)	0.388(3)	0.435(2)	**0.467**(1)
F1$^+$	0.364(4)	0.390(3)	0.432(2)	**0.469**(1)

+(-): the higher (smaller) the value, the better the performance.

Table 4.10: Number of instances in the initial and edited training datasets.

	Initial training data	Edited training data
Emotions	391	113
Yeast	1500	832
Medical	645	624
Enron	1123	861
Webpage	22000	13693

Table 4.11: Running Time (in Seconds) on the test sets for the two methods.

	EMLkNN		Rank-SVM	
	Before Editing	After Editing	Before Editing	After Editing
Emotions	1.4	0.4	162.9	9.9
Yeast	12.4	10.1	$1.1 * 10^4$	$0.2 * 10^4$
Medical	7.3	3.7	$1.6 * 10^4$	$1.4 * 10^4$
Enron	18.8	8.2	$1.0 * 10^4$	$1.6 * 10^4$
Webpage	61.7	47.7	$2.1 * 10^4$ s $\simeq$ 14 h	$8.6 * 10^3$ s $\simeq$ 5.9 h

Table 4.12: Experimental results (mean±std) on the Emotions dataset.

	EMLkNN		Rank-SVM	
	Before Editing	After Editing	Before Editing	After Editing
Hamming loss$^-$	0.1906 ± 0.0197●	0.1457 ± 0.0641	0.1918 ± 0.0217●	0.1481 ± 0.0474
One-Error$^-$	0.2664 ± 0.0438●	0.1881 ± 0.1000	0.2560 ± 0.0806●	0.1580 ± 0.0691
Coverage$^-$	1.8160 ± 0.1985○	1.5238 ± 0.4024	1.7022 ± 0.2927○	1.5049 ± 0.4016
Ranking loss$^-$	0.1734 ± 0.0278○	0.1310 ± 0.0612	0.1564 ± 0.0396●	0.1000 ± 0.0495
Average precision$^+$	0.7997 ± 0.0300●	0.8642 ± 0.0544	0.8069 ± 0.0437●	0.8742 ± 0.0477
Accuracy$^+$	0.5578 ± 0.0453●	0.6810 ± 0.1199	0.5409 ± 0.0495●	0.6582 ± 0.0946
Precision$^+$	0.6882 ± 0.0515●	0.7742 ± 0.0948	0.6629 ± 0.0651●	0.7566 ± 0.0805
Recall$^+$	0.6414 ± 0.0511●	0.7717 ± 0.0862	0.6542 ± 0.0625●	0.7681 ± 0.1017
F1$^+$	0.6352 ± 0.0453●	0.7479 ± 0.0951	0.6260 ± 0.0519●	0.7348 ± 0.0850

●(○): statistically significant (non-significant) difference of performance of the classification algorithm applied on the initial and the edited dataset, based on two-tailed paired t-test at 5% significance.

Table 4.13: Experimental results (mean±std) on the Yeast dataset.

	EMLkNN		Rank-SVM	
	Before Editing	After Editing	Before Editing	After Editing
Hamming loss$^-$	0.2012 ± 0.0100●	0.1681 ± 0.0410	0.1946 ± 0.0061●	0.1574 ± 0.0384
One-Error$^-$	0.2416 ± 0.0268●	0.1704 ± 0.0744	0.2156 ± 0.0320○	0.1652 ± 0.0724
Coverage$^-$	6.4813 ± 0.2632●	5.7267 ± 0.8003	6.3363 ± 0.2360●	5.4958 ± 0.7416
Ranking loss$^-$	0.1862 ± 0.0148●	0.1393 ± 0.0533	0.1652 ± 0.0073●	0.1164 ± 0.0451
Average precision$^+$	0.7574 ± 0.0181●	0.8131 ± 0.0588	0.7727 ± 0.0126●	0.8291 ± 0.0587
Accuracy$^+$	0.5241 ± 0.0213●	0.6036 ± 0.0738	0.5291 ± 0.0163●	0.6110 ± 0.0763
Precision$^+$	0.6820 ± 0.0243●	0.7342 ± 0.0601	0.6951 ± 0.0163●	0.7552 ± 0.0571
Recall$^+$	0.6136 ± 0.0229●	0.7051 ± 0.0757	0.6331 ± 0.0220●	0.7125 ± 0.0795
F1$^+$	0.6214 ± 0.0210●	0.6964 ± 0.0683	0.6336 ± 0.0114●	0.7074 ± 0.0692

●(○): statistically significant (non-significant) difference of performance of the classification algorithm applied on the initial and the edited dataset, based on two-tailed paired t-test at 5% significance.

Table 4.14: Experimental results (mean±std) on the Medical dataset.

	EML*k*NN		Rank-SVM	
	Before Editing	After Editing	Before Editing	After Editing
Hamming loss$^-$	0.0168 ± 0.0031○	0.0149 ± 0.0016	0.0106 ± 0.0016○	0.0095 ± 0.0014
One-Error$^-$	0.2769 ± 0.0679○	0.2415 ± 0.0283	0.1370 ± 0.0333○	0.1181 ± 0.0338
Coverage$^-$	3.3559 ± 1.1640○	4.1286 ± 0.8808	1.0705 ± 0.3695○	0.8641 ± 0.2841
Ranking loss$^-$	0.1078 ± 0.0343●	0.1513 ± 0.0273	0.0138 ± 0.0069○	0.0111 ± 0.0050
Average precision$^+$	0.7840 ± 0.0471○	0.7957 ± 0.0186	0.9055 ± 0.0220○	0.9151 ± 0.0213
Accuracy$^+$	0.5923 ± 0.0638●	0.6420 ± 0.0339	0.7194 ± 0.0421●	0.7690 ± 0.0296
Precision$^+$	0.6536 ± 0.0654●	0.7049 ± 0.0332	0.7614 ± 0.0496●	0.8097 ± 0.0344
Recall$^+$	0.6109 ± 0.0612●	0.6675 ± 0.0375	0.7573 ± 0.0410●	0.8186 ± 0.0321
F1$^+$	0.6192 ± 0.0632●	0.6718 ± 0.0342	0.7462 ± 0.0438●	0.7994 ± 0.0300

●(○): statistically significant (non-significant) difference of performance of the classification algorithm applied on the initial and the edited dataset, based on two-tailed paired t-test at 5% significance.

Table 4.15: Experimental results (mean±std) on the Enron dataset.

	EML*k*NN		Rank-SVM	
	Before Editing	After Editing	Before Editing	After Editing
Hamming loss$^-$	0.0621 ± 0.0104○	0.0569 ± 0.0038	0.0558 ± 0.0135○	0.0485 ± 0.0027
One-Error$^-$	0.5715 ± 0.0978●	0.3903 ± 0.0746	0.2937 ± 0.1114○	0.2146 ± 0.0488
Coverage$^-$	25.6451 ± 6.3233○	23.2417 ± 2.4229	13.9366 ± 3.2339●	11.4764 ± 1.1672
Ranking loss$^-$	0.3345 ± 0.0863○	0.2939 ± 0.0575	0.0978 ± 0.0237●	0.0719 ± 0.0124
Average precision$^+$	0.4673 ± 0.0563○	0.5772 ± 0.0523	0.6140 ± 0.0871●	0.7020 ± 0.0347
Accuracy$^+$	0.1646 ± 0.0582●	0.3659 ± 0.0477	0.3797 ± 0.1249●	0.4857 ± 0.0398
Precision$^+$	0.3343 ± 0.1058●	0.5416 ± 0.0592	0.5748 ± 0.0968●	0.6603 ± 0.0378
Recall$^+$	0.1836 ± 0.0663●	0.4212 ± 0.0541	0.4660 ± 0.1097●	0.5985 ± 0.0437
F1$^+$	0.2185 ± 0.0746●	0.4477 ± 0.0539	0.4940 ± 0.1049●	0.6021 ± 0.0402

●(○): statistically significant (non-significant) difference of performance of the classification algorithm applied on the initial and the edited dataset, based on two-tailed paired t-test at 5% significance.

Conclusions

This thesis addressed the problem of multi-label classification, where each instance can be assigned to one or more than one class. In recent years, multi-label learning is increasingly demanded by real-world applications, such as text categorization, scene analysis, bioinformatics and music classification. For example, an electronic document related to sports can also be labeled as economic. Several methods have been proposed in the literature to solve the multi-label learning problem. These methods may be influenced by the quality of training instances, and their characteristics.

The goal of this thesis was twofold:

– Designing classifiers for multi-label learning that can handle uncertainties presented in the training dataset;
– Making the training dataset more reliable using an edition algorithm.

In the rest, we briefly summarize our contributions and then discuss some future works.

Summary and contributions

Chapter 1 of this thesis discussed the existing methods for multi-label learning. We have shown that there exist three main categories depending on the learning process from the training datasets: problem transformation, algorithm adaptation and ensemble methods. Methods in the first category transform the multi-label problem into different problems of single-label learning while having computational complexity to model some characteristics of multi-label learning. Methods in the second category extend the existing traditional classifiers to the framework of multi-label learning. In general, these methods perform better than those of the first category at the price of higher complexity. The third category seeks to reduce the previous problems and to take into account the advantages of the two previous categories. Also in Chapter 1, we have presented some applications and introduced some evaluation metrics for multi-label learning.

Chapter 2 reviewed the basics of belief functions theory, which is a general framework for reasoning with uncertainty. This chapter described extensions of this theory to lattices in order to reduce the complexity of belief functions. We showed how this formalism can be used to represent uncertainty about set-valued variables, as required in multi-label learning tasks.

In Chapter 3, we have presented some characteristics of multi-label problem and focused on correlations that can exist between labels. We have shown such a correlation in two real-world datasets. The principle of the *RAKEL* approach was extended to the framework of belief functions theory in order to handle uncertainty that may exist in the training set. The resulting approach, referred to as *Evidential RAKEL*, incorporates two stages: inducing many classifiers on different small subsets of labels, and then proceeding to gather all information given by the different classifiers in order to combine them under the theory of evidence. Two ways to make a final decision regarding the predicted label set for an example have been explained: the *bel* function which aims to predict the output for each label separately, and the maximum of communality which is determined via solving an integer programming problem under constraints.

An illustration on a synthetic dataset showed the ability of our method to handle uncertainty in the training set. Experimental results on real-world datasets show also that *Evidential RAKEL* tends to outperform the classical *RAKEL* method in terms of *predictive-based* metrics.

An important contribution of this thesis is the editing algorithm for multi-label learning introduced in Chapter 4. The goal of this approach is to edit (retain/remove) each instance in the training set in order to get a subset of more reliable instances. As a result, training a multi-label classifier becomes easier in practice. Our editing algorithm is based on the voting of all neighbors for an instance and on a simple criteria, the Hamming loss, to estimate the classification error of the target instance.

An illustration on a synthetic dataset showed that our editing algorithm could remove outliers and erroneous instances, and made us able to get smoother decision boundaries between classes. Two multi-label classification methods (EMLkNN and Rank-SVM) were applied to a large set of real-world data after editing. When compared to the same methods on datasets before editing, learning from reduced datasets was shown to have the following advantages:

– It achieves better predictive performance, according to *prediction-based* and *ranking-based* metrics;
– It has lower running time;
– It reduces the memory space to store training instances.

Future Work

Further extensions and improvements can be considered as a continuation of this work.

The *Evidential RAKEL* method demonstrated good predictive performance on two datasets selected from the library of multi-labeled data. It will be interesting to confirm the efficiency of this method on other challenging real-world applications, after the space complexity limitation has been overcome. Future work will consider the development of a toolbox for the theory of evidence on large sets to implement evidential reasoning without limitation on CPU time.

This thesis showed that learning from edited data improves the performances of learning algorithm and reduces the running time. Future research should consider applying the existing algorithm to other applications domains and with other classifiers to investigate the merit of editing in these settings. We could also study other editing mechanisms based on other approaches, like support vector machines or neural networks.

It will also be interesting to explore the editing idea with imbalanced data, in which examples from some classes outnumber examples from other classes. The motivation is to detect erroneous or mislabeled instances without influencing the multi-label characteristic of the given dataset. This suggests that instances, belonging to classes with few examples should remain unchanged or should be edited carefully.

Another direction for future research is to adapt condensing approaches in the domain of multi-label learning. These approaches would select the minimal consistent subset of instances that represents all possible classes and their relationships. This idea could accelerate the response time of multi-label classifiers and further reduce their computational complexity, by learning from a more coherent set.

Bibliography

[1] Aha, D. W., Kibler, D., and Albert, M. K. Instance-based learning algorithms. *Machine Learning 6*, 1 (Jan. 1991), 37–66. 79

[2] Aly, M. Survey on Multi-Class Classification Methods. Tech. rep., Caltech, USA, 2005. 13

[3] Angiulli, F. Fast condensed nearest neighbor rule. In *Proceedings of the 22nd international conference on Machine learning - ICML '05* (Bonn, Germany, 2005), vol. IT-14, ACM Press, pp. 25–32. 78

[4] Ayache, N. Epidaure: a research project in medical image analysis, simulation, and robotics at INRIA. *IEEE transactions on medical imaging 22*, 10 (Oct. 2003), 1185–201. 25

[5] Barbedo, J. G. A., and Lopes, A. Automatic Genre Classification of Musical Signals. *EURASIP Journal on Advances in Signal Processing 2007*, 1 (2007), 064960. 81

[6] Bay, S. D. Combining Nearest Neighbor Classifiers Through Multiple Feature Subsets. In *Proceedings of the 17th International Conference on Machine Learning* (Madison, WI, 1998), pp. 37–45. 13

[7] Bishop, C. M. *Neural Networks for Pattern Recognition.* Oxford University Press, Inc., 1995. 13

[8] Boutell, M. R., Luo, J., Shen, X., and Brown, C. M. Learning multi-label scene classification. *Pattern Recognition 37*, 9 (Sept. 2004), 1757–1771. 1, 14, 66

[9] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. *Classification and Regression Trees.* Wadsworth Publishing Company, 1984. 67

[10] Brighton, H., and Mellish, C. Advances in Instance Selection for Instance-Based Learning Algorithms. *Data Mining and Knowledge Discovery 6*, 2 (2002), 153–172. 75

[11] Brodley, C. E., and Friedl, M. A. Identifying mislabeled training data. *Journal Of Artificial Intelligence Research 11* (1999), 131–167. 73, 74, 77

[12] Brucker, F., Benites, F., and Sapozhnikova, E. Multi-label classification and extracting predicted class hierarchies. *Pattern Recognition 44*, 3 (Mar. 2011), 724–738. 15

[13] Chang, C.-l. Finding Prototypes For Nearest Neighbor Classifiers. *IEEE Transactions on Computers C-23*, 11 (Nov. 1974), 1179–1184. 78

[14] Chapelle, O., Schölkopf, B., and Zien, A. *Semi-supervised learning.* MIT Press, Cambridge, MA, 2006. 11

[15] Chen, K., Gao, S., Zhu, Y., and Sun, Q. Music Genres Classification using Text Categorization Method. In *2006 IEEE Workshop on Multimedia Signal Processing* (Oct. 2006), IEEE, pp. 221–224. 24

[16] Chen, W., Yan, J., Zhang, B., Chen, Z., and Yang, Q. Document Transformation for Multi-label Feature Selection in Text Categorization. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)* (Omaha, NE, Oct. 2007), IEEE, pp. 451–456. 1, 14

[17] Chevaleyre, Y., and Zucker, J. Solving multiple-instance and multiple-part learning problems with decision trees and rule sets. Application to the mutagenesis problem. In *the 14th Biennial Conference of the Canadian Society for Computational Studies of Intelligence* (Ottawa, Canada, 2001), Springer-Verlag, London, UK, pp. 204–214. 15

[18] Clare, A., and King, R. D. Knowledge Discovery in Multi-Label Phenotype Data. In *Proceedings of the 5th European Conference on Principles of Data*

Mining and Knowledge Discovery (PKDD '01) (Baden-Wurttemberg, Germany, 2001), vol. 2168, Springer-Verlag, London, UK, pp. 42–53. 20, 21

[19] Cobb, B. R., and Shenoy, P. P. On the plausibility transformation method for translating belief function models to probability models. *International Journal of Approximate Reasoning 41*, 3 (Apr. 2006), 314–330. 35

[20] Cover, T., and Hart, P. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory 13*, 1 (Jan. 1967), 21–27. 74

[21] Dasarathy, B. V. Minimal consistent set (MCS) identification for optimal nearest neighbor decision systems design. *IEEE Transactions on Systems, Man, and Cybernetics 24*, 3 (Mar. 1994), 511–517. 79

[22] de Carvalho, A., and Freitas, A. A. A Tutorial on Multi-label Classification Techniques. In *Foundations of Computational Intelligence Volume 5*, Studies in Computational Intelligence. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 177–195. 73

[23] De Comité, F., Gilleron, R., and 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)* (Leipzig, Germany, June 2003), Lecture Notes in Computer Science, Springer-Verlag, Berlin, Heidelberg, pp. 35–49. 21

[24] Denœ ux, T. Conjunctive and disjunctive combination of belief functions induced by nondistinct bodies of evidence. *Artificial Intelligence 172*, 3 (Feb. 2008), 234–264. 34

[25] Denoeux, T., and Masson, M.-H. Evidential reasoning in large partially ordered sets. *Annals of Operations Research 195*, 1 (May 2011), 135–161. 2, 7, 31, 37, 42, 57, 58, 59

[26] Denoeux, T., Younes, Z., and Abdallah, F. Representing uncertainty on set-valued variables using belief functions. *Artificial Intelligence 174*, 7-8 (May 2010), 479–499. 2, 7, 31, 37, 39, 42, 43, 55, 83

[27] Devijver, P. A. On the editing rate of the Multiedit algorithm. *Pattern Recognition Letters 4*, 1 (Feb. 1986), 9–12. 76

[28] Dey, T. K., Janoos, F., and Levine, J. A. Meshing interfaces of multi-label data with Delaunay refinement. *Engineering with Computers 28*, 1 (Apr. 2011), 71–82. xv, 25

[29] Dietterich, T. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence 89*, 1-2 (Jan. 1997), 31–71. 14, 15

[30] Dubois, D., and Prade, H. Representation and combination of uncertainty with belief functions and possibility measures. *Computational Intelligence 4*, 3 (Sept. 1988), 244–264. 34

[31] Elisseeff, A., and Weston, J. Kernel methods for Multi-labelled classification and Categorical regression problems. In *Advances in Neural Information Processing Systems 14* (2001), vol. 14, Biowulf Technologies, MIT Press, pp. 681–687. 14, 20, 21, 84

[32] Fayed, H. a., and Atiya, A. F. A novel template reduction approach for the K-nearest neighbor method. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council 20*, 5 (May 2009), 890–6. 79

[33] Fu, G., Nan, X., Liu, H., Patel, R. Y., Daga, P. R., Chen, Y., Wilkins, D. E., and Doerksen, R. J. Implementation of multiple-instance learning in drug activity prediction. *BMC bioinformatics 13 Suppl 1*, Suppl 15 (Jan. 2012), S3. 14

[34] Fung, G., Dundar, M., Bi, J., and Rao, B. A fast iterative algorithm for fisher discriminant using heterogeneous kernels. In *Twenty-first international conference on Machine learning (ICML '04)* (Banff, Alberta, Canada, 2004), ACM Press, p. 40. 13

[35] García, S., Derrac, J., Cano, J. R., and Herrera, F. Prototype Selection for Nearest Neighbor Classification: Survey of Methods. Tech. rep., 2010. 75

[36] García, S., Derrac, J., Cano, J. R., and Herrera, F. Prototype selection for nearest neighbor classification: taxonomy and empirical study. *IEEE transactions on pattern analysis and machine intelligence 34*, 3 (Mar. 2012), 417–35. 8, 73, 74, 75

[37] Gates, G. The reduced nearest neighbor rule (Corresp.). *IEEE Transactions on Information Theory 18*, 3 (May 1972), 431–433. 78

[38] Glover, F., and Laguna, M. *Tabu Search.* Springer US, Boston, MA, 1997. 80

[39] Godbole, S., and Sarawagi, S. Discriminative methods for multi-labeled classification. In *Proceedings of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2004)* (Sydney, Australia, 2004), Springer Berlin Heidelberg, pp. 22–30. 21

[40] Grabisch, M. Belief functions on lattices. *International Journal of Intelligent Systems 24*, 1 (Jan. 2009), 76–95. 37, 38

[41] Grabisch, M., and Labreuche, C. Bi-capacities Part I: Definition, Möbius Transform and Interaction. *Fuzzy Sets and Systems 151*, 2 (2005), 211–236. 40

[42] Guan, D., Yuan, W., Lee, Y.-K., and Lee, S. Nearest neighbor editing aided by unlabeled data. *Information Sciences 179*, 13 (June 2009), 2273–2282. 77

[43] Hadsell, R., Chopra, S., and LeCun, Y. Dimensionality Reduction by Learning an Invariant Mapping. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2 (CVPR'06)* (2006), vol. 2, IEEE, pp. 1735–1742. 11

[44] Hattori, K., and Takahashi, M. A new edited k-nearest neighbor rule in the pattern classification problem. *Pattern Recognition 33*, 3 (Mar. 2000), 521–528. 76

[45] Ho, S.-Y., Liu, C.-C., and Liu, S. Design of an optimal nearest neighbor classifier using an intelligent genetic algorithm. *Pattern Recognition Letters 23*, 13 (Nov. 2002), 1495–1503. 80

[46] JIANG, Y., AND ZHOU, Z.-H. Editing Training Data for kNN Classifiers with Neural Network Ensemble. *Lecture Notes in Computer Science 3173* (2004), 356–361. 77

[47] JIN, B., MULLER, B., ZHAI, C., AND LU, X. Multi-label literature classification based on the Gene Ontology graph. *BMC bioinformatics 9* (Jan. 2008), 525. 25

[48] KANJ, S., ABDALLAH, F., AND DENŒ UX, T. Evidential Multi-label Classification Using the Random k-Label Sets Approach. In *Proceedings of the 2nd Int. Conf. on Belief Functions* (Compiègne, France, 2012), Springer, AISC 164, pp. 21–28. 8

[49] KANJ, S., ABDALLAH, F., AND DENŒ UX, T. La méthode RAkEL évidentielle pour la classification multi-label. In *Rencontres Francophones sur la Logique Floue et ses Applications (LFA 2012)* (Compiègne, France, 2012), Cépaduès-Editions, pp. 185–192. 8

[50] KANJ, S., ABDALLAH, F., AND DENOEUX, T. Purifying training data to improve performance of multi-label classification algorithms. In *Proceedings of the 15th Int. Conf. on Information Fusion (FUSION 2012)* (Singapore, 2012), IEEE, pp. 1784–1792. 8

[51] KHAN, S. S., AND MADDEN, G. M. A Survey of Recent Trends in One Class Classification. In *Proceedings of the 20th Irish Conference on Artificial Intelligence and Cognitive Science (AICS '09)* (Dublin, Ireland, 2009), Springer Berlin Heidelberg, pp. 188—-197. 13

[52] KOPLOWITZ, J., AND BROWN, T. A. On the relation of performance to editing in nearest neighbor rules. *Pattern Recognition 13*, 3 (Jan. 1981), 251–255. 76

[53] KOTSIANTIS, S. B., ZAHARAKIS, I. D., AND PINTELAS, P. E. Machine learning: a review of classification and combining techniques. *Artificial Intelligence Review 26*, 3 (Nov. 2007), 159–190. 11

[54] KUNCHEVA, L. I. Editing for the k-nearest neighbors rule by a genetic algorithm. *Pattern Recognition Letters 16*, 8 (Aug. 1995), 809–814. 80

[55] Lewis, D. D., Yang, Y., Rose, T. G., and Li, F. RCV1: A New Benchmark Collection for Text Categorization Research. *Journal of Machine Learning Research 5* (2004), 361–397. 24

[56] Li, Y., Hu, Z., Cai, Y., and Zhang, W. Support vector based prototype selection method for nearest neighbor rules. In *Proceedings of the first international conference on Advances in Natural Computation* (Changsha, China, 2005), Springer Berlin Heidelberg, pp. 528–535. 73, 81

[57] López, V. F., de la Prieta, F., Ogihara, M., and Wong, D. D. A model for multi-label classification and ranking of learning objects. *Expert Systems with Applications 39*, 10 (Aug. 2012), 8878–8884. 14

[58] Lukashevich, H., Abeber, J., Dittmar, C., and Grossmann, H. From multi-labeling to multi-domain-labeling: A novel two-dimensional approach to music genre classification. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR '09)* (Kobe, Japan, 2009), International Society for Music Information Retrieval, pp. 459–464. 24

[59] Luo, X., and Zincir-heywood, A. N. Evaluation of Two Systems on Multiclass Multi-label Document Classification. In *Proceedings of the 15th International Symposium on Methodologies for Intelligent Systems (ISMIS'05)* (Saratoga Springs, NY, 2005), Springer-Verlag Berlin, pp. 161–169. 22

[60] Madjarov, G., Kocev, D., Gjorgjevikj, D., and Džeroski, S. An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition 45*, 9 (Sept. 2012), 3084–3104. xv, 1, 6, 14, 18, 73

[61] Monjardet, B. The presence of lattice theory in discrete problems of mathematical social sciences. Why. *Mathematical Social Sciences 46*, 2 (Oct. 2003), 103–144. 37, 38

[62] Nair, P. C. Content Based Automatic Categorization and Annotation of Medical Images. *International journal of Computing Technology and Information Security 1*, 2 (2011), 35–39. 25

[63] Pavlidis, P., and Grundy, W. N. Combining microarray expression data and phylogenetic profiles to learn gene functional categories using support vector machines. Tech. rep., Department of Computer Science, Columbia University, New York, 2000. 92

[64] Pestian, J. P., Brew, C., Matykiewicz, P., Hovermale, D. J., Johnson, N., Cohen, K. B., and Duch, W. A Shared Task Involving Multi-label Classification of Clinical Free Text. In *Proceedings of the Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing (BioNLP '07)* (Prague, Czech Republic, 2007), vol. 1, Association for Computational Linguistics, pp. 97–104. 5, 81, 92

[65] Pkalska, E., Duin, R. P., and Paclík, P. Prototype selection for dissimilarity-based classifiers. *Pattern Recognition 39*, 2 (Feb. 2006), 189–208. 73

[66] Qi, G.-J., Hua, X.-S., Rui, Y., Tang, J., Mei, T., and Zhang, H.-J. Correlative multi-label video annotation. In *Proceedings of the 15th international conference on Multimedia - MULTIMEDIA '07* (Augsburg, Germany, 2007), ACM Press, p. 17. 1, 14

[67] Quinlan, J. R. *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers Inc., 1993. 13

[68] Quost, B., Denoeux, T., and Masson, M. Pairwise classifier combination in the transferable belief model. In *Proceedings of the 7th International Conference on Information Fusion (Fusion '05)* (Philadelphia, PA, USA, 2005), IEEE, p. 8. 13

[69] Read, J. A pruned problem transformation method for multi-label classification. In *Proceedings of the New Zealand Computer Science Research Student Conference (NZCSRSC '08)* (Christchurch, New Zealand, 2008), pp. 143–150. 19

[70] Read, J. *Scalable Multi-label Classification.* PhD thesis, University of Waikato, Hamilton, New Zealand, Feb. 2010. 50, 51

[71] Read, J., Pfahringer, B., Holmes, G., and Frank, E. Classifier chains for multi-label classification. *Machine Learning 85*, 3 (June 2011), 333–359. 5, 19, 23, 73

[72] Rousu, J., Saunders, C., Szedmak, S., and Shawe-Taylor, J. Kernel-Based Learning of Hierarchical Multilabel Classification Models. *Journal of Machine Learning Research 7* (2006), 1601–1626. 15

[73] Sánchez, J., Pla, F., and Ferri, F. Prototype selection for the nearest neighbour rule through proximity graphs. *Pattern Recognition Letters 18*, 6 (June 1997), 507–513. 73

[74] Sanden, C., Befus, C. R., and Zhang, J. Perception based Multi-genre Labeling on Music Data. In *Proceedings of the International Computer Music Conference (ICMC)* (Stony Brook, NY, USA, 2010), pp. 9–15. 24

[75] Savage, L. J. *Foundations of Statistics.* John Wiley, New York, 1954. 35

[76] Schapire, R. E., and Singer, Y. Improved Boosting Algorithms Using Confidence-rated Predictions. *Machine learning 37*, 3 (1999), 297–336. 82

[77] Schapire, R. E., and Singer, Y. BoosTexter : A Boosting-based System for Text Categorization. *Machine Learning 39*, 2-3 (2000), 135–168. 21

[78] Schwarz, E. M. Genomic classification of protein-coding gene families. *WormBook : the online review of C. elegans biology* (Jan. 2005), 1–23. 25

[79] Sebastiani, F. Machine learning in automated text categorization. *ACM Computing Surveys 34*, 1 (Mar. 2002), 1–47. 24

[80] Shafer, G. *A Mathematical Theory of Evidence.* Princeton University Press, 1976. 2, 32, 33

[81] Shen, X., Boutell, M., Luo, J., and Brown, C. Multi-Label Machine Learning and Its Application to Semantic Scene Cassification. In *Proceedings of Storage and Retrieval Methods and Applications for Multimedia* (San Jose, CA, USA, Dec. 2003), M. M. Yeung, R. W. Lienhart, and C.-S. Li, Eds., vol. 5307, SPIE, pp. 188–199. 1, 14

[82] Shetty, J., and Adibi, J. The Enron Email Dataset Database Schema and Brief Statistical Report. Tech. rep., Information Sciences Institute Technical Report, University of Southern California, 2004. 92

[83] Silla, C. N., and Freitas, A. A. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery 22*, 1-2 (Apr. 2011), 31–72. 15

[84] Smets, P., and Kennes, R. The transferable belief model. *Artificial Intelligence 66*, 2 (Apr. 1994), 191–234. 31, 35

[85] Stanevski, N., and Tsvetkov, D. Using Support Vector Machine as a Binary Classifier. In *Proceedings of the International Conference on Computer Systems and Technologies (CompSysTech' 05)* (Varna, Bulgaria, 2005), no. 2, pp. 1–5. 13

[86] Swonger, C. W. Sample set condensation for a condensed nearest neighbor decision rule for pattern recognition. In *Frontiers of Pattern Recognition*, S. Watanabe, Ed. Academic Press, 1972, pp. 511–519. 78

[87] Tahir, M. A., Kittler, J., and Bouridane, A. Multilabel classification using heterogeneous ensemble of multi-label classifiers. *Pattern Recognition Letters 33*, 5 (Apr. 2012), 513–523. 6, 23, 24, 68

[88] Tang, L., Rajan, S., and Narayanan, V. K. Large scale multi-label classification via metalabeler. In *Proceedings of the 18th international conference on World wide web (WWW '09)* (Madrid, Spain, 2009), ACM Press, p. 211. 92

[89] Thabtah, F. A., Cowling, P., and Peng, Y. MMAC: A new multi-class, multi-label associative classification approach. In *Proceedings of the 4th IEEE International Conference on Data Mining (ICDM '04)* (Bradford, UK, 2004), IEEE, IEEE Computer Society, pp. 217–224. 22

[90] Tomek, I. Two Modifications of CNN. *IEEE Transactions on Systems, Man, and Cybernetics 6*, 11 (1976), 769–772. 76, 78

[91] Trohidis, K., Tsoumakas, G., Kalliris, G., and Vlahavas, I. Multilabel classification of music into emotions. In *Proceedings of the 9th International*

Conference on Music Information Retrieval (ISMIR '08) (Philadelphia, PA, USA, 2008), pp. 325–330. 66, 92

[92] Tsoumakas, G., Dimou, A., Spyromitros, E., Mezaris, V., Kompatsiaris, I., and Vlahavas, I. Correlation-based pruning of stacked binary relevance models for multi-label learning. In *Proceedings of the Workshop on Learning from Multi-Label Data (MLD'09)* (Bled, Slovenia, 2009), pp. 101–116. 19

[93] Tsoumakas, G., and Katakis, I. Multi-Label Classification : An Overview. *International Journal of data warehousing & mining 3*, 3 (2007), 1–13. 1, 8, 22, 26

[94] Tsoumakas, G., Katakis, I., and Vlahavas, I. Mining Multi-label Data. In *Data Mining and Knowledge Discovery Handbook*, O. Maimon and L. Rokach, Eds. Springer US, Thessaloniki, Greece, 2010, pp. 667–685. 27, 73, 82

[95] Tsoumakas, G., Katakis, I., and Vlahavas, I. Random k-Labelsets for Multilabel Classification. *IEEE Transactions on Knowledge and Data Engineering 23*, 7 (July 2011), 1079–1089. 2, 6, 23, 48, 49, 68

[96] Tsoumakas, G., and Vlahavas, I. Random k-Labelsets : An Ensemble Method for Multilabel Classification. In *the 18th European Conference on Machine Learning* (Warsaw, Poland, 2007), Springer Berlin Heidelberg, pp. 406–417. 48, 49

[97] Van Hulse, J., and Khoshgoftaar, T. Knowledge discovery from imbalanced and noisy data. *Data & Knowledge Engineering 68*, 12 (Dec. 2009), 1513–1542. 73

[98] Vateekul, P. *Hierarchical Multi-Label Classification : Going Beyond Generalization Trees.* PhD thesis, University of Miami, 2012. 15

[99] Veloso, A., Jr, W. M., Gonc, M., and 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)* (Berlin, Heidelberg, 2007), no. 20060519184000, Springer-Verlag, pp. 605–6012. 22

[100] Vens, C., Struyf, J., Schietgat, L., Džeroski, S., and Blockeel, H. Decision trees for hierarchical multi-label classification. *Machine Learning 73*, 2 (Aug. 2008), 185–214. 15

[101] Verbaeten, S., and Assche, A. V. Ensemble Methods for Noise Elimination in Classification Problems. In *Proceedings of the 4th international conference on Multiple classifier systems (MCS'03)* (Guildford, UK, 2003), vol. 2709, Springer-Verlag Berlin, Heidelberg, pp. 317–325. 77

[102] Vishwanathan, S., and Murty, N. Use of Multi Category Proximal SVM for Dataset Reduction. In *Proceedings of the 1st International Workshop on Hybrid Intelligent Systems* (Adelaide, Australia, 2001), M. Köppen, Ed., Hybrid Information Systems, pp. 19–24. 80

[103] Wang, J., and Zucker, J.-D. Solving multiple-instance problem: A lazy learning approach. In *Proceedings of the 17th International Conference on Machine Learning* (San Francisco, CA, 2000), no. 1994, pp. 1119–1125. 15

[104] Wang, L., Chang, M., and Feng, J. Parallel and sequential support vector machines for multi-label classification. *International Journal of Information Technology 11*, 9 (2005), 11–18. 21

[105] Wilson, D. L. Asymptotic Properties of Nearest Neighbor Rules Using Edited Data. *IEEE Transactions on Systems, Man, and Cybernetics 2*, 3 (1972), 408–421. 76

[106] Wilson, D. R., and Martinez, T. R. Reduction Techniques for Instance-Based Learning Algorithms. *Machine learning 38*, 3 (2000), 257–286. 80

[107] Wu, X., Kumar, V., Ross Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Yu, P. S., Zhou, Z.-H., Steinbach, M., Hand, D. J., and Steinberg, D. Top 10 algorithms in data mining. *Knowledge and Information Systems 14*, 1 (Dec. 2007), 1–37. 74

[108] Wu, Y., Ianakiev, K., and Govindaraju, V. Improved k-nearest neighbor classification. *Pattern Recognition 35*, 10 (Oct. 2002), 2311–2318. 79

[109] Xu, J. An extended one-versus-rest support vector machine for multi-label classification. *Neurocomputing 74*, 17 (Oct. 2011), 3114–3124. 5, 21, 73

[110] Xu, J. An efficient multi-label support vector machine with a zero label. *Expert Systems with Applications 39*, 5 (Apr. 2012), 4796–4804. 21, 96

[111] Younes, Z., Abdallah, F., Denoeux, T., and Snoussi, H. A Dependent Multilabel Classification Method Derived from the -Nearest Neighbor Rule. *EURASIP Journal on Advances in Signal Processing 2011*, 1 (2011), 645964. 1, 5, 14, 22, 50, 73, 83

[112] Yu, G., Domeniconi, C., Rangwala, H., Zhang, G., and Yu, Z. Transductive multi-label ensemble classification for protein function prediction. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '12)* (New York, New York, USA, 2012), ACM Press, p. 1077. 5, 25

[113] Zhang, H., and Sun, G. Optimal reference subset selection for nearest neighbor classification by tabu search. *Pattern Recognition 35*, 7 (July 2002), 1481–1490. 80

[114] Zhang, M.-l. A k-Nearest Neighbor Based Multi-Instance Multi-Label Learning Algorithm. In *Proceedings of the 22nd IEEE International Conference on Tools with Artificial Intelligence* (Arras, Oct. 2010), IEEE, pp. 207–212. 15

[115] Zhang, M.-L., and Zhou, Z.-h. ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognition 40*, 7 (July 2007), 2038–2048. 20, 22, 67, 83

[116] Zhou, Z.-H., Zhang, M.-L., Huang, S.-J., and Li, Y.-F. Multi-instance multi-label learning. *Artificial Intelligence 176*, 1 (Jan. 2012), 2291–2320. 15

[117] Zhu, J., and Lu, L. Perceptual Visualization of a Music Collection. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME '05)* (Amsterdam, The Netherlands, 2005), IEEE, pp. 1058–1061. 24