Combination robust to dependence between classifiers in a decentralized learning context

Mahmoud Albardan

Université de Lille

CRIStAL
Centre de Recherche en Informatique, Signal et Automatique de Lille

École doctorale 072 : Sciences Pour l'Ingénieur (EDSPI)

Combinaison robuste à la dépendance entre classifieurs dans un contexte d'apprentissage décentralisé

THÈSE

Présentée pour obtenir le grade de docteur délivré par

l'Université de Lille

Spécialité AGITSI: Automatique, Génie Informatique, Traitement du Signal et des Images

soutenue publiquement le 11 Octobre 2018 par

Mahmoud ALBARDAN

devant le jury composé de:

Laurent Heutte	Professeur, Université de Rouen	Rapporteur
Sylvie le Hégarat	Professeur, Université Paris-Sud	Rapportrice
Didier Coquin	Professeur, Université Savoie Mont Blanc	Examinateur
Benjamin Guedj	Chargé de recherche, Inria	Examinateur
John Klein	Maître de conférences, Université de Lille	Co-directeur
Olivier Colot	Professeur, Université de Lille	Directeur

Université de Lille
Centre de Recherche en Informatique, Signal et Automatique de Lille (CRIStAL)
UMR CNRS 9189, 59650 Villeneuve-d'Ascq, France.

Acknowledgements

This thesis was achieved within the Sigma team - CRIStAL laboratory in the University of Lille under the supervision of Prof. *Olivier Colot* and Prof. *John Klein* from October 2015 to October 2018.

First of all, i am very grateful to my Creator who helped me to achieve this thesis and make it as good as possible.

I would like to thank the jury members for accepting to judge this work and to the effort they made to provide advices and propositions in order to improve this work.

I would like to thank all my colleagues in the Sigma team for the discussions we had and for the tips they gave to make this work better before starting writing this manuscript.

A big thanks also to all my colleagues in the P2-building for the discussions we had and certainly for the very friendly environment they provided during the past three years.

I am very thankful to my supervisors Olivier Colot and John Klein for accepting me as their PhD student, believing in me to complete this mission and helping and guiding me along these three years.

Infinite particular thanks go to my supervisor John Klein for his permanent presence during these three years, his guidance, advices and efforts that he provided in order to complete this work.

I would like also to thank all my friends in France and in Lebanon. I am also very grateful to my family, my parents who supported me infinitely during the years i spent inside and outside the academic life, my brother Ahmad and my two sisters Nour and Lynn.

October 2018
Mahmoud

Abstract

Machine learning is a rapidly growing field both in the number of the employed methods and also in the number of databases available to users and their usages. Classification, which is a task mainly addressed by machine learning, is thus affected by this evolution. For example, the problem of decentralized learning (data divided into distinct subsets on networked nodes) encourages the creation of global systems based on classifier ensembles. For this reason, this thesis addresses the problem of multi-classifier systems or classifier ensembles. The goal of my research is then the design of multi-classifier systems that provide a certain level of robustness (performance at least equivalent to the maximally accurate classifier of the system but in a context of wider usage). To this end, we propose two solutions that are the main contributions of the thesis. The first one is a possibilistic approach based on a combination of possibility distributions computed from confusion matrices using a Tnorm function, while the second is a probabilistic approach based on a combination of conditional probabilities using a Copula function. By setting the unique hyperparameter of each of these by grid search, we are able to either explicitly capture some of the statistical dependency relationship between the predictions of the individual classifiers or to circumvent the hypothesis of independence between the latter which is a key point in obtaining the desired robustness.

Résumé

L'apprentissage automatique est un domaine en forte croissance à la fois dans le nombre de méthodes employées mais aussi dans le nombre de bases de données à disposition des utilisateurs et dans les usages de ces derniers. La classification, qui est une tâche essentiellement abordée par l'apprentissage automatique, est ainsi affectée par cette évolution. Par exemple, la problématique de l'apprentissage décentralisé (données réparties en sous-ensembles distincts sur des noeuds en réseau) incite à créer des systèmes globaux basés sur des comités de classifieurs. Pour cette raison, on traite dans cette thèse la problématique des systèmes multi-classifieurs ou bien les comités de classifieurs. L'objectif de mes travaux de recherche est alors la conception des systèmes multi-classifieurs qui assurent un certain niveau de robustesse (performances au moins équivalente au meilleur classifieur du comité mais dans un contexte d'utilisation plus large). A cette fin, nous proposons deux solutions qui sont les principales contributions de la thèse. La première est une approche possibiliste basée sur une combinaison des distributions de possibilité calculées à partir des matrices de confusion à l'aide d'un fonction Tnorme, tandis que la deuxième est une approche probabiliste basée sur une combinaison de probabilités conditionnelles à l'aide d'une fonction Copule. En réglant l'unique hyperparamètre de chacun de ces fonctions de combinaison par recherche par quadrillage (grid search), nous sommes capables soit de capturer explicitement une partie de la relation de dépendance statistique entre les prédictions des classifieurs du comité soit de contourner l'hypothèse d'indépendance entre ces derniers ce qui est un point clé dans l'obtention de la robustesse souhaitée.

Table of contents

List of figures

List of tables

List of Symbols

m	number of classes
Ω	The set of classes or the frame of discernment
c_i	The i^{th} element in Ω
c_{rej}	The rejection class
$\mathbf{x}$	An input vector
y	The label of $\mathbf{x}$
d	The dimension of the input vector
$\hat{c}$	The decision and the prediction function
X, Y	Random variables of the input and the class labels respectively
$\mathbf{W}$	A weight matrix
$\mathbf{b}$	A bias vector
Φ	The mapping function for dimension transformation
$\boldsymbol{\theta}$	A set of parameters
$J(\boldsymbol{\theta})$	The cost function for logistic regression algorithm
$\mathscr{D}_{\text{train}}$	Training set
$\mathscr{D}_{\text{val}}$	Validation set
n_{train}	Number of samples in $\mathscr{D}_{\text{train}}$
n_{val}	Number of samples in $\mathscr{D}_{\text{val}}$

sigm	Sigmoid function
H	Hidden layer in the neural net
O	Output layer in the neural net (Softmax)
los	The cross entropy loss function for neural network
Reg	Regularizer
K	Number of classifiers
M	Confusion matrix of a classifier
T_λ	The parametrized t-norm function
$\hat{\lambda}$	Estimated parameter
$\mathscr{C}$	The copula function
$\mathscr{W}$	Minimal bound of the copula function
$\mathscr{M}$	Maximal bound of the copula function
G	Vector of cumulative marginal distributions
F	Cumulative joint distribution
c	Density of the copula function

List of Acronyms

AI	Artificial intelligence
MCS	Multiple Classifier System
SVM	Support Vectors Machine
CNN	Convolutional Neural Network
k-nn	k-nearest neighbors
LDA	Linear Discriminant Analysis
QDA	Quadratic Discriminant Analysis
Prec	Precision of a classifier
Rec	Recall of a classifier
Acc	Accuracy of a classifier
Sensi	Sensitivity of a classifier
Speci	Specificity of a classifier
ROC	Receiver Operating Characteristic
AUC	Area Under Curve
EoC	Ensemble of Classifiers
BKS	Behavior Knowledge Space
RSM	Random Subspaces Method
ECOC	Error Correcting Output Codes
MCMC	Markov chain Monte Carlo

General introduction

Artificial intelligence (AI) is the collection of theories and techniques that allow to simulate human intelligence using machines. It involves different concepts such as extracting information from data in order to learn (i.e. discover patterns), fusing information and making decisions. A philosophical debate is still open about the name −artificial intelligence− since the term «intelligence» involves different concepts such as creativity or emotion which a humanly programmed machine does not possess. Also, the term «artificial» conveys the idea that what the machine does is not real. A widely accepted understanding of the terminology is the following: an artificially intelligent machine interacts with its environment and makes autonomous decisions as if it were intelligent (but these seemingly autonomous decisions are just the consequences of cleverly written code lines). In spite of this controversy, the influence of artificial intelligence on today's society and on the future of human kind has been growing dramatically in the past few years due to recent research developments. It is at present time a major political issue and several governments have started to address specifically this question. As other previous powerful technologies did (e.g. steam engines, electricity, computers), artificial intelligence will have a strong impact on human society in everyday life domains such as economics and finance, security, human relations and interactions, education, healthcare, transports and so on.

In the economical field, artificial intelligence is modifying economical models at national or international scales because of the growing role that play intelligent machines in the production line. In the wake of Industry 4.0, manufacturers are now able to deliver high quality services and products, where high precision and special care is demanded and some plants are operational with minimal human supervision. But artificial intelligence impact is not limited to industry. For example, it is used in agriculture as well to predict diseases in agricultural parcels by taking advantages of features such as meteorological conditions. Artificial intelligence has been also deeply rooted in finance and banking for over a decade. Many banks and financial consulting companies benefit from the advantages that offer a group of computerized tools (algorithms) in artificial intelligence. For instance, artificial

intelligence is used to anticipate the risk that a client withdraws his financial funds from a bank, or to detect frauds in card payment transactions.

In our daily lives, artificial intelligence has also already been having a significant impact in human interactions especially with the growth of the number of users of social networks such as Facebook, Twitter, Whatsapp and other similar applications. When these latter emerged, these networks were meant only to facilitate communication between people (from potentially different nations and different backgrounds). Yet, the economical model of these companies involve the exploitation of users' activities which are recorded and saved to build very large databases. Such a large amount of data lead to scientific challenges which are now referred to as «big data». Social network users now receive personalized advertisement and are automatically identified in pictures they upload. Of course, artificial intelligence in daily life is not limited to social networks. We now have personal assistants which can be commanded by voice and self-driving cars are entering our streets.

In spite of the advantages artificial intelligence may offer (automation of tedious tasks, increased safety and efficiency), it can be argued that it also comes with a number of risks if it is not closely regulated and supervised by legal means or control agencies. Recently, two use cases of artificial intelligence have been highlighted in this scope. The first one deals with autonomous cars but also applies to more general situations. Indeed, what decision must the car make if it must choose between two options each of which may be fatal to either the driver or to a nearby pedestrian? There already are reported fatal car accidents involving an autonomous car. This is both an ethical and legal matter that is unresolved. It is not even clear who should be held reponsible for such a tragic situation (the car as legal person, the car company, the programmer or the car owner because he or she agreed the terms by the time the car was bought). The second recently debated topic is the exploitation of artificial intelligence by the armament industry and many companies are now committing themselves not to develop such applications.

Without denying the importance and the relevance of these debates, we envisage artificial intelligence only for the scientific challenges it raises in terms of data analysis and processing as part of this PhD. Just like many other past scientific contributions, this is up to governments to make sure that these contributions are utilized in an ethical and safe way. In this thesis, we are interested in the development of machine learning which is at the core of artificial intelligence. Machine learning is about the usage of statistical and computerized techniques to allow the computer to «learn» and discover patterns in the data that might be valuable for a specific application. Machine learning involves solving many types of problems such as supervised, unsupervised, reinforcement, semi-supervised and active learning problems.

In supervised learning, the learner is given a set of example/answer pairs and must identify a mapping from the space where example live to the set of possible answers. When these possible answers are categories, one speaks of a classification task. When the answers live in a continuum of values, one speaks of a regression task. For instance, the examples may be vectors depicting a bank customer with entries like age, income, indebtedness, and so on. If the bank tries to learn a function that predicts if a loan should be granted or not, then this is classification. If the bank tries to learn a function that predicts which amount of money can be loaned, then this is regression.

In the unsupervised setting, one must solve the same problem from examples (inputs) only. The answers to which these examples must be mapped are not known and one must discover patterns from input distribution solely. It is consequently, in general, a more challenging class of problems. When the set of answers are categories, the task that consists in finding the mapping from examples to answers is called cluster analysis or clustering in which similar objects are pooled in a cluster. Another task that received considerable attention in the past decade is the derivation of recommender systems. These systems try typically to suggest a number of items to a website user given his/her previous interest in other items and given the history of other users who had interest in the same items. The algorithms allowing to build a recommender system share a number of aspects from both supervised and unsupervised techniques but it is a research topic of its own.

A new type of machine learning algorithms that has gained popularity in the last few years is reinforcement learning. Reinforcement learning concepts are inspired from animal psychology and were later applied in automatic and computer science. According to Sutton and Barto [93], reinforcement learning is about learning to take decisions (or actions) given its current state by maximizing a quantifiable *reward* signal. This is the main difference with supervised learning where the data is presented to the algorithm in terms of (input, label) pairs. However, in the former, there is no label and we just provide to the algorithm the obtained reward following the chosen action. The learning process is done by iteratively trying sequences of actions and getting rewards until the integrated reward signal is maximized. A popular reinforcement learning framework is the Markovian decision process setting where states are random variables and state transitions following decisions (i.e. actions) are also random. This is convenient when one must learn in an uncontrolled environment. A welcoming playground for these types of algorithms are games with planification issues. For instance, Deepmind developed an algorithm called AlphaGo based on reinforcement learning that is capable to play Go game. In 2017, AlphaGo defeated the Go game world champion.

Semi-supervised learning is a category of algorithms that are middle ground between supervised and unsupervised learning. The algorithms of this category use a small amount of labeled data (supervised) and a large amount of unlabeled data (unsupervised). In general, the algorithm benefits of unlabeled data to achieve better prediction performances than the one obtained if only the labeled data subset is used to train the model. Ideally, we would like the semi-supervised algorithm to achieve a level of performance close to a supervised algorithm if we knew the labels of each data point. A simple method to perform semi-supervised learning is *pseudo-labeling* which is divided in three steps. First, we train the model on the small amount of labeled data then we use the trained model to predict the labels of the unlabeled data. Finally, we re-train the model on the whole dataset (with known or estimated labels) as in supervised learning. The main motivation of such algorithms is the difficulty of labeling a large amount of data because it demands a human expert intervention, or a special equipment which is costly and tedious.

Active learning is quite similar to semi-supervised learning in the sense that it uses a bunch of labeled and unlabeled data. The idea of active learning is to allow the algorithm to decide which training sample it wants to be labeled in order to be trained on. In other words, an algorithm inspects first an amount unlabeled data, then it *requests* which samples should be labeled. After that, an expert annotates (labels) the selected samples (by the algorithm) and sends it back to the algorithm for a supervised training session. This process (algorithm training)$\longleftrightarrow$(expert labeling) is then iteratively repeated until we obtain the final model. In spite of its clear definition, active learning is not trivial. The main difficulty lies in how the algorithm selects (or not) a sample to be labeled by the expert. In active learning, a current model selects the most informative samples to be labeled by the expert. This is called the query selection strategy. There exist many of them such as uncertainty sampling and query by committee. In the former, the current model selects samples for which it is the most uncertain about. For instance, the current model can be tested on the unlabeled data and samples for which the model is the least certain are selected. In the latter, an ensemble of classifiers are trained on the current labeled data then applied on the unlabeled data. The samples that create the highest level of disagreement in the ensemble are selected.

In this manuscript, our interest is limited to supervised learning and specifically to classification. When a learner, i.e. a training algorithm, is run on data it produces a prediction function that maps examples to the finite set of answers. This function is also called a classifier. The aim of this thesis is to achieve robust classification with respect to performances suhc as the classification accuracy which is the proportion of correctly classified objects over the total number of objects to be classified. Several types of robustnesses can be sought:

- robustness to data distribution variability: one can try to derive a method that is suboptimal but always significantly better than random guess. Such a method could be used as reference as compared to near optimal models that need to be built for each task specifically.

  A second justification for such methods is that sometimes, one cannot make assumptions on some parameters or aspects of the classification problem. For example, in these seminal work, we are interested in learning from decentralized datasets. If $\mathscr{D}$ denotes a matrix whose columns are the training examples (as column vectors), there are many situations in which a learner will have access to a limited number of lines or a limited number of columns of $\mathscr{D}$. If we can use several learners trained from a partition of the data, we need to make the corresponding classifier collaborate to merge their predictions efficiently.

- robustness to parameter tuning: a robust learner performances should not change drastically when the parameters of the algorithm change softly.

To derive robust classification methods, we propose to investigate and build Multiple Classifier Systems (MCS) in this PhD.

A multiple classifier system (MCS) is composed of at least two classifiers whose individual responses about an object are fused to elaborate a final combined decision about the label that will be assigned to the object. Once trained, multiple classifiers provide multiple learning functions which offer, if they are efficiently combined, more learning capacity to the final model and introduce more options on the class label of an object. However, this advantage depends on the presence of *diversity* in the learned functions which is crucial to build an MCS with powerful discriminative performances.

A common definition of diversity relates to the differences in predictions that a multiple classifier system can produce but there is no a formal precise definition of diversity. Diversity can be induced in several ways but the decentralized setting is, in general, a favorable context for diversity because the classifiers are trained on different data points or features. However, training a set of diverse classifiers does not achieve statistical independence of the classifier outputs. The level of dependency between classifier predictions is highly variable and has its roots in the distribution of the data as well as on the capacities of the training algorithms. For instance, in the decentralized setting, if we split data points uniformly at random and apply the same training algorithms on each data subset, then we will obtain highly dependent predictions. If we split data points so that data subsets are in disjoint regions of the feature space, then the predictions will be far less dependent. Also, dependency is difficult to

measure using a single statistic. For example, it is well known that Pearson's coefficient can grasp linear dependence but fails to characterize other forms of dependence. In this work, we examine parametric models of dependency between classifiers and propose classifier combination techniques that take into consideration the hidden relations between classifiers and adapt itself to it.

Let alone dependency, classifier fusion method should be able to adjust the impact of each classifier on the final decisions. Intuitively, contextual information such as classifier individual performance estimates are relevant features to build a combination rule. Indeed, such information express the degree of confidence that can be given to a classifier's decision and we should downweight those classifiers with poor performances.

Contextual performances or contextual data might be given by several individual performance criteria of the classifiers computed on a validation set (a subset of training examples set aside to that end). Typical criteria are accuracy, recall, precision, area under ROC curve or compound ones such as the F-measure. These criteria differ in the amount and the nature of the provided information about a classifier performance. Most of these measures are derived from confusion matrices which encapsulate a finer granularity on a classifier ability to yield correct predictions. Also known as error matrix, it provides information about the classification performances observed on the validation set. The size of a confusion matrix is $m \times m$ where m is the number of labels. The rows refer to the true labels and the columns refer to the predicted labels. In a confusion matrix, an element of coordinate (i, j) is the number of samples whose true label is the i^{th} class but were assigned to the j^{th} class by the classifier.

A principled way to aggregate classifier is to estimate the distribution of the true class label given the classifier predictions. This distribution is related to the conditional distributions of each classifier prediction given the true class. We can derive (frequentist) estimates of these latter distributions from confusion matrices. This probabilistic combination does not necessarily require probabilistic classifiers. Note that the larger the validation set is, the more accurate are the estimations of these probability distributions.

Summarizing the above paragraphs, the originality of these seminal works lies in the ability of the proposed classifier combination methods to adapt itself to classifier dependency while taking into account their individual performances to achieve robust fusion. We also want to the combination process to have (at most) polynomial complexity in the both the number of possible class labels and the number of aggregated classifiers.

This manuscript is organized in 3 chapters. Following the general introduction, the first chapter is dedicated to a review of the state of the art in classifier combination. We

first present some common classification algorithms and the motivations behind classifier combinations. Then we present a wide range of combination techniques including some well known algorithms such as voting, Borda counts and ensemble methods. We also present trainable fusion algorithms that learn the combination rule and finally we discuss fusion approaches in different uncertainty frameworks.

In the second chapter, we present the first contribution of this thesis. As previously mentioned, the goal of this PhD is to derive robust multiple classifier systems that are able to manage the dependency between classifiers and make use of contextual data in the combination algorithm. Conditional probability distributions computed from confusion matrices are good candidates as contextual data since they allow to determine which classifiers perform well and have tractable memory complexity. Using Bayes theorem and under conditional independence assumptions, these distributions allow the inference of the posterior probability through which a label is assigned to an unseen test sample. To circumvent those unrealistic independence assumptions, we propose to carry over the problem to the possibilistic framework in which a consensus possibility distribution can be obtained by invoking combination rules that are computationally tractable and do not resort to a parameter space with a very high dimension. The possibility theory is an alternative to probability theory as uncertainty framework. It is highly coupled to a form of modal logic in which propositions are assigned a graded epistemic state of truth. If a proposition is known to be true then we are totally confident in that proposition as if it had been assigned probability one. However, if a proposition is not known to be true, then it does not mean that it is false. In a probabilistic version of the model, we would only be able to say that the probability of the proposition belongs to the unit interval. In this framework, we investigate a t-norm based combination rule family. This family is parametrized by a hyperparameter that is tuned via grid search. This method is tested on public datasets and compared to a benchmark of other combination approaches. A statistical study is carried out to validate these experimental results. A classifier fault tolerance study entails the chapter in order check the sensitivity of each combination method in the benchmark to noisy information.

In the third chapter, we present another approach to combine classifiers. In contrast to the t-norm based possibilistic method proposed in chapter 2, this approach does not demand any problem transfer to another uncertainty framework but instead performs the combination within the probabilistic framework. Under 0-1 loss, the optimal classification rule based on classifier outputs consists in maximizing the probability of the true class label given the predicted class labels issued by the classifiers. The evaluation of all these conditional distributions has an exponential complexity in the number of classifiers. To circumvent

this problem, we propose to apply Bayes rule and to compute the joint distribution out of marginal distributions obtained from confusion matrices using a *copula* function which is a probabilistic model that capture the dependence between random variables which are classifier outputs in our situation. In this chapter, we examine two types of copulas: the parametrized Gaussian copula and the independent copula. The latter is equivalent to making conditional independence assumptions so that the joint distribution is the product of the marginals and we retrieve a well known combination scheme. The parameters of the Gaussian copula are entries of a correlation matrix. We investigate only a subset of these possible matrices so that there is only one parameter to tune to regulate the level of dependence among classifier outputs (like in the possibilistic approach). We also carry a number of numerical experiments on public datasets to assess the relevance of this Gaussian copula model. In this third chapter, we focus on experiments in a decentralized setting where each learner has access to a subset of training data points. A comparison with the possibilistic approach is proposed.

Finally, we entail this manuscript with a conclusion where we comment on the contributions of these seminal works and enumerate several interesting perspectives and future research directions.

Chapter 1

State-of-the art on classifier combination

1.1 Introduction

Classification is a supervised machine learning task consisting of assigning objects (inputs) to discrete categories (classes). A very large literature is devoted to algorithmic solutions for this task. Building a single robust classifier is challenging because classification is an ill-posed problem in the sense that one is trying to retrieve a prediction function from only a finite set of noisy input-output pairs known as the training set. Multiple classifiers have been introduced to provide different levels of capacity[1] and robustness making them more or less appropriate to a given dataset and also to a given application. For instance, Convolutional Neural Networks (CNNs) are performing very well on large datasets containing input signals such as images [63] but random forests or SVMs are still achieving state-of-the-art performances on datasets in which inputs are categorical entries [39],[74]. Consequently, there is no general solution to all classification problems, as a result known from the famous «no free lunch theorem».

As the arsenal of classification algorithms increased dramatically, it became more and more tempting to use several classifiers and then combine their decisions to gain in accuracy and avoid the burden of choosing the right one. Note that a combination of classifiers remains itself a classifier and the no free lunch theorem also applies to it. There is no hope for a universally optimal classifier but one can achieve other forms of robustness that we will later discuss. One of the simplest fusion methods is to perform majority voting on the set of predicted classes delivered by each classifier.

In this thesis, we focus on classifier fusion at the decision (or output) level. Fusion is indeed

[1] The capacity of a model learned by an algorithm corresponds to the intuition of the size of the functional space in which we pick the prediction function.

also possible at intermediate stages. For instance, several features can be extracted from the original inputs using supervised or unsupervised methods. The set of features is concatenated into an input feeding another classifier. This can be particularly useful in multi-task learning where a first task (face pose recognition) learns useful representations for another task (facial landmark detection) [107]. More generally, computing a new representation of input examples from a set of raw features is known as *feature fusion*.
A relevant categorization of classifier output natures is found in [101] where the authors identify three natures of output information:

- elementary outputs (the output of the classifier is a single label),
- ranked outputs (the output of the classifier is a list of labels ranked from most probable to least probable),
- and scored outputs (the classifier assigns a degree of confidence to each class label).

It is possible to switch to the first (respectively second) nature from the third one by selecting the class with highest score (respectively by ranking class labels according to their score). In case of tied scores, this may no longer be possible but it may be useful to perform reject in these circumstances which means to give a classifier the possibility to recommend not to classify a sample because of an excessive level of uncertainty. Thus scored outputs are more informative and consequently many classifiers provide score-based outputs. For the sake of adaptability, a classifier combination approach should be designed for elementary outputs and we are in line with this requirement in the approaches introduced as part of this thesis and that will be presented in chapter 2 and 3.
In this chapter, we first present the classification problem in section 1.2 and a collection of the most widely used supervised learning algorithms where some of which will serve as base classifier for combination. In section 1.3 we explain the motivations of our work and give some generalities about classifier combination. In sections 1.5,1.6, 1.7 and 1.8 we review a wide range of combination schemes. We start with classical combination techniques and then we review some widely used ensemble methods such as bagging and boosting. In section 1.7 we present another category of fusion algorithms that learn combination parameters or rules such as stacking, mixture models and mixture of experts. Afterwards, we review classifier combination methods under uncertainty, i.e. probabilistic, evidential and fuzzy frameworks. Finally we entail this chapter with a conclusion in which we position the contributions of this thesis with respect to the state-of-the-art.

1.2 Classification problem statement

In this section, we explain in detail some learning algorithms that were employed in our proposed combination algorithms as a part of this thesis. In particular, we present some classical classification methods such as support vector machines, k-nearest neighbors, discriminant analysis classifiers and neural networks. We also present decision trees, naive Bayes classifier and logistic regression that are used in the experiments involving the proposed combination approaches. Even though we only used probabilistic classifiers (in the sense that they output conditional probabilities) in these experiments, the proposed combination methods in this thesis are not restricted to this category of learners but can be also applied on deterministic classifiers.

1.2.1 Notations

Let Ω denote a set of m class labels $\Omega = \{c_1,..,c_m\}$ and c_{rej} denote that the artificial tag assigned to test samples rejected by a classifier in case this latter is trained with a reject option. Each c_i represents one label or class and Ω is the set of all labels. Let $\mathbf{x}$ denote an input example with d entries. Most of the time, $\mathbf{x}$ is a vector and lives in $\mathbb{R}^d$ but sometimes some of its entries are categorical data and $\mathbf{x}$ lives in an abstract space $\mathbb{X}$ which does not necessarily have a vector space structure. Without loss of generality, we suppose that $\mathbf{x}$ is a vector in the rest of this manuscript unless stated otherwise.

A classification problem consists in determining a prediction function $\hat{c}$ that maps any input $\mathbf{x}$ with its actual class $y \in \Omega$. This function is obtained from a training set $\mathscr{D}_{\text{train}}$ which contains pairs $\left(\mathbf{x}^{(i)}, y^{(i)}\right)$ where $y^{(i)}$ is the class label of example $\mathbf{x}^{(i)}$. Given a classifier, the label y assigned to the input $\mathbf{x}$ is denoted by $\hat{c}(\mathbf{x})$.

As explained in the introduction, the predictive function $\hat{c}(\mathbf{x})$ may be written as a composition between a decision function and the classifier output. Indeed, if the classifier produces ranked outputs, the decision function should select the top one class. When classifiers rely on discriminative or generative models (*e.g.* logistic regression, naive Bayes, *etc.*), the outputs are predictive probabilities of the classes given the input $p(Y = c_i|X = \mathbf{x})$ where X is the random variable capturing the variability in the inputs and Y the random variable capturing the variability in the classes. In this case, a result from decision theory states that selecting the class with the highest predictive probability minimizes the 0-1 expected loss. Note that a probabilistic aggregation of classifiers does not necessarily require that classifiers are also themselves probabilistic therefore we do not make such an assumption in what follows.

1.2.2 Classification algorithms

In this subsection, we present some supervised learning algorithms. According to the nature of outputs, classifiers can be categorized into probabilistic, which produce scored outputs, and deterministic which do not produce scored outputs. For instance, decision trees, neural networks, logistic regression, naive Bayes and discriminant analysis classifiers are all probabilistic since they produce predictive conditional probabilities of class labels given the input while support vector machines and k-nearest neighbors are deterministic since they produce a decision (along with a score) and they are not originally built on any probabilistic framework. Note that k-nearest neighbors classifier has been extended to the probabilistic framework [75],[48],[97] in contrary to support vector machines classifiers which do not have any probabilistic counterpart.

Supervised classification methods can be also categorized into parametric and non-parametric according to the model that they rely on. Non parametric algorithms such as k-nearest neighbors and decision trees do not make any explicit assumptions about the model underlying the mapping from inputs to class labels while parametric algorithms do make such assumptions. For example, naive Bayes, logistic regression, discriminant analysis classifiers and neural networks are parametric algorithms and the class of functions that can learned by them is captured by a vector of parameters. Besides, being non parametric does not mean the absence of *model hyperparameters* which adjust the learning process in contrary to *model parameters* which allow the model to predict. For instance, the weight matrice entries in a neural network are parameters of the model while the learning rate and the number of hidden layers are hyperparameters. To recap, a non parametric algorithm may have hyperparameters such as k-nearest neighbors whose hyperparameter is the number of neighbors k.
Before introducing classification algorithms, we give the definitions of two important notions in machine learning *overfitting* and *regularization*:

- Overfitting: this phenomenon occurs when a classifier learns the function that fit exactly the training data but does not generalize well in the sense that unseen examples at traning time are not well classified. The main reason behind overfitting is the lack of a sufficient amount of training data especially when the model involves a large number of parameters to learn.

- Regularization: this technique is one of the solutions to mitigate overfitting. It consists in penalizing some values of the parameters and therefore it controls the flexibility of the model which reduces the risk of overfitting the training data.

The above notions will be mentioned later in the presentation of the classifiers In the rest of this subsection, we present 7 different classifiers:

k-nearest neighbors classifier

k-nearest neighbors (*k*-nn) is a simple classification algorithm that assigns labels to new instances based on a metric (figure 1.1). The algorithm finds the k closest (in the sense of the metric in question) training samples to a new instance $\mathbf{x}$ and then performs a majority voting on the k training labels of the neighbors to predict the label of $\mathbf{x}$. Usual metrics are Euclidean, Minkowski or Mahalanobis distances [19]. Beside being simple to implement and not requiring an explicit training phase, *k*-nn does not demand any parameter learning and it is based uniquely on local information. However, the test phase can be computationally consuming as the computation cost of the metric is polynomial in the dimensionality of the inputs and one needs to compute n_{train} distances. In a big data context, *k*-nn is thus not recommended.

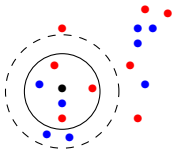


Fig. 1.1 Illustration of *k*-nn algorithm. A simple majority vote assigns the test sample (black dot) to the red label for $k = 5$ (solid line circle). However, if $k = 7$ (dashed line circle) the sample will be assigned to the blue label.

Figure 1.1 gives an illustration of the principle of the *k*-nn algorithm.

Decision trees

Decision trees are graphical models belonging to early machine learning approaches. Similarly to other algorithms, they may be used for both regression and classification tasks, where they are called regression and classification trees respectively. In a decision tree, each node represents a feature (attribute), each branch represents a decision to make based on a feature value as compared to a threshold and each leaf is class label. Decision trees are built sequentially by splitting the training set into *pure* subsets, e.g. subsets having the same label on the leaves, with respect to the features. In other words, we learn a partition of the features space in which each feature space region is a hypercube. A famous algorithm to construct a decision tree is **ID3** which defines a splitting rule (which attribute to split on and

as compared to which threshold) based on purity of the resulting subsets using Shanon's entropy and information gain. Probabilities involved in entropy computation are calculated frequentistically for each subset. Information gain fuses then entropy measures for all subsets under every node. The attribute having the highest information gain is selected to split on. Decision trees are not computationally expensive since the cost is logarithmic in the number data points used during the construction. Besides they are simply understood and interpreted. On the other hand, they are prone to overfitting especially when dealing with high dimensional feature spaces. In spite of its easy construction, a small variation in learning samples may yield a completely different tree but fortunately a decision tree ensemble can insure a certain level of stability. For instance, Brieman [7] studied the effect of Bagging (explained later section 1.6) in reducing instability and concluded that the efficiency of bagging might be more clear if the classifiers are instable. Similarly, Skurichina [91] conducted a large number of experiments to stabilize classifiers using bagging.

Example 1. Suppose a 2-dimensional dataset describing a car weight (heavy, light) and horsepower (high, low). The aim is to predict whether the car has a high or low mileage. During training, an algorithm defines which feature is the next to split on. In this tree, the nodes are the features, the branches are possible feature values and the leaves are class labels (see Figure 1.2 for an illustration).

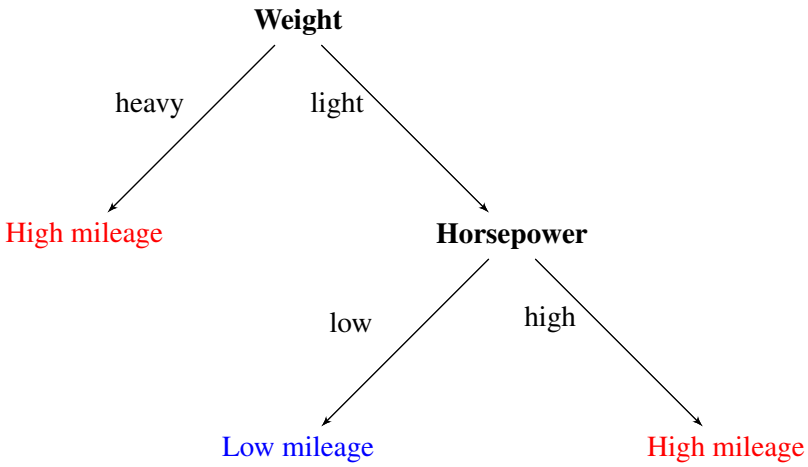


Fig. 1.2 Illustration of a learned decision tree. The first split is done on the feature weight and the second split is done on a feature horsepower. The leaves represent the class labels (binary).

Naive Bayesian classifier

The naive Bayesian classifier is a probabilistic classifier whose decision rule is based on Bayes theorem. It «naively» assumes that all features describing a target value are independent

given that target value. However, this assumption is not true in reality since features might be correlated because they describe the same object. Considering a d-dimensional input vector $\mathbf{x} = (x_1, .., x_d)$ and y a given target value (label), by applying Bayes theorem, we have:

$$\begin{aligned} p(Y = y|\mathbf{x}) &= \frac{p(y).p(\mathbf{x}|Y = y)}{p(\mathbf{x})}, && (1.1)\\ &= \frac{p(y).p(x_1, .., x_d|Y = y)}{p(x_1, .., x_d)}, && (1.2)\\ &\propto p(y).p(x_1, .., x_d|Y = y). && (1.3) \end{aligned}$$

Assuming class conditional independence between all entries of $\mathbf{x}$ given y, the posterior probability is given by:

$$p(Y = y|\mathbf{x}) \propto p(y)\prod_{i=1}^{d} p(x_i|Y = y). \quad (1.4)$$

The decided label is then the one that maximizes $p(Y = y|\mathbf{x})$:

$$\hat{c}(\mathbf{x}) = \arg\max_{y\in\Omega} \; p(y)\prod_{i=1}^{d} p(x_i|Y = y). \quad (1.5)$$

If we do not have any prior on the class distribution *p(y)* a frequentest computation can be used on the learning dataset to estimate these latter. To estimate each attribute distribution $p(x_i|Y = y)$, one has to propose a model and estimate its parameters. For binary and discrete feature values Bernoulli and Multinomial models are natural choices because they are the canonical distributions induced by the sampling procedure. In the continuous case, a classical hypothesis is that data are normally distributed. Thus, a Gaussian distribution may be fitted for each feature and each class label. Since proving the efficiency of a combination method is our main goal, we are not much interested in deriving very accurate models. Thus, we assume that features are normally distributed in each experiment involving a naive Bayes classifier.

Support vector machines

Support vector machines (SVMs) are deterministic models used for supervised binary classification tasks. In its simplest form, SVM are considered as linear classifiers since they build a separating (*d-1*)-dimensional hyperplane $\mathscr{S}$ to assign labels to d-dimensional input vectors. Extensions to more than two class settings are possible by resorting to an ensemble method

that combines several such hyperplanes. Typical policies consist in building «one versus all» or «one versus one» base classifiers. SVM learning function is computed by maximizing the distance between the hyperplane and the closest training input vector for each label (Figure 1.3). These training inputs are called support vectors and the minimal distance in question is called the margin.

Let us assume that we address a binary classification problem where an input vector $\mathbf{x}$ has entries living in $\mathbb{R}^d$, and the true label y of $\mathbf{x}$ belongs to the set $\{-1,+1\}$. A hyperplane $\mathscr{S}$ can be written as:

$$\mathbf{w}^t \cdot \mathbf{x} + b = 0 \tag{1.6}$$

where $\mathbf{w}$ is the normal vector to $\mathscr{S}$, $\frac{|b|}{||\mathbf{w}||}$ is the distance from the hyperplane to the origin and $(\cdot)$ the dot product. We define here two specific hyperplanes (figure 1.3) passing through the closets points in each class and their equations can be reformulated as follows:

$$\mathscr{S}_1 : \mathbf{w}^t \cdot \mathbf{x} + b = +1 \implies \text{Points on or above } \mathscr{S}_1 \text{ are assigned to } y_1 = +1 \tag{1.7}$$

$$\mathscr{S}_2 : \mathbf{w}^t \cdot \mathbf{x} + b = -1 \implies \text{Points on or below } \mathscr{S}_2 \text{ are assigned to } y_2 = -1 \tag{1.8}$$

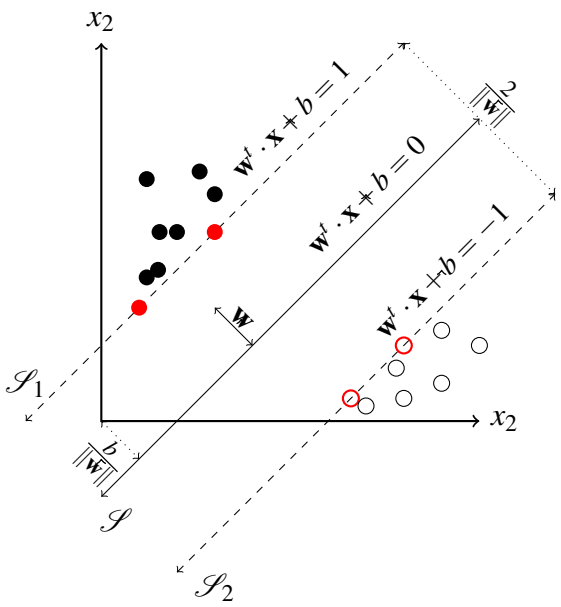


Fig. 1.3 Illustration of a linear SVM for a binary classification problem. (Figure originally coded in Latex by Yifan Peng and slightley modified [yif])

The learning procedure of an SVM relies on the maximization of the *margin*. Regardless of the value of b, the margin is always equal to $\frac{2}{||\mathbf{w}||}$ and one aims to maximize this distance such that $\mathscr{S}$ «comfortably» separates data points of class y_i for $i \in \{1,2\}$ thus two conditions must be respected:

$$\left.\begin{array}{ll} \mathbf{w}\cdot\mathbf{x}+b \geq +1 & \text{if y} = +1 \\ \mathbf{w}\cdot\mathbf{x}+b \leq +1 & \text{if y} = -1 \end{array}\right\} \Longrightarrow y(\mathbf{w}^t\cdot\mathbf{x}+b) \geq 1. \tag{1.9}$$

Therefore, the optimization problem is: Maximizing $\frac{2}{||\mathbf{w}||}$ subject to $y(\mathbf{w}^t\cdot\mathbf{x}+b) \geq 1$. Obviously, the previous formulation is limited to the case where data points are linearly separable but fortunately the extension of the method to non linearly separable datasets is not problematic. We need to introduce slack variables which will quantify to what extent a given example violates the margin. The cost function is modified so that the sum of those violations remains reasonably small.

When dataset are not overlapping but the optimal frontier is not linear, another very nice extension consists in projecting input vectors into a new (usually of higher dimension than the original space) feature space where they are easily discriminated using a linear classification rule. This embedding is done via the following mapping:

$$\Phi : \mathbb{R}^d \longrightarrow \mathbb{R}^{d'} s.t.\ \ d' \geq d \tag{1.10}$$

Learning a classification rule in $\mathbb{R}^{d'}$ might be computationally expensive since it requires to compute the transformation Φ and then perform the operations in $\mathbb{R}^{d'}$. However, there is a way to overcome this issue known as the *kernel trick*. At both training and test time, the SVM setting requires only to compute the inner products $\Phi(\mathbf{x})^t\cdot\Phi(\mathbf{x}')$ for some $\mathbf{x}$ and $\mathbf{x}' \in \mathbb{R}^d$. One can define a function Kern such that:

$$\text{Kern}(\mathbf{x},\mathbf{x}') = \Phi(\mathbf{x})^t\cdot\Phi(\mathbf{x}'). \tag{1.11}$$

This function is called a *kernel*. The advantage of this trick is that the computation of Kern is moderate, even if computing images of inputs through Φ are costly. A popular example of kernels is the radial basis function for which it can be proved that the corresponding Φ function for this kernel send inputs to an infinite dimensional space.

Linear and Quadratic discriminant analysis classifiers

Discriminant analysis classifiers (LDA and QDA) are probabilistic generative models. In such a model, a model of the joint distribution of $(\mathbf{X},Y)$ is learned. A test sample $\mathbf{x}$ is assigned to the class label with the highest predictive probability $p(Y=y|\mathbf{x})$:

$$\hat{c}(\mathbf{x}) = \arg\max_{y\in\Omega} \; p(Y=y|\mathbf{x}) \tag{1.12}$$

$$= \arg\max_{y\in\Omega} \; \frac{p(\mathbf{x}|Y=y)\times p(Y=y)}{p(\mathbf{x})} \tag{1.13}$$

For a binary classification problem $\Omega=\{0,1\}$, discriminant analysis classifiers assume that conditional probabilities $p(\mathbf{x}|Y=y)$ have a multivariate normal distribution with density:

$$p(\mathbf{x}|Y=0) = \frac{1}{(2\pi)^{\frac{d}{2}}|\varepsilon_0|^{\frac{1}{2}}}\exp(-\frac{1}{2}(\mathbf{x}-\mu_0)^t\cdot\varepsilon_0^{-1}\cdot(\mathbf{x}-\mu_0)), \tag{1.14}$$

$$p(\mathbf{x}|Y=1) = \frac{1}{(2\pi)^{\frac{d}{2}}|\varepsilon_1|^{\frac{1}{2}}}\exp(-\frac{1}{2}(\mathbf{x}-\mu_1)^t\cdot\varepsilon_1^{-1}\cdot(\mathbf{x}-\mu_1)), \tag{1.15}$$

where (μ_i,ε_i) are the mean vector of size d and the covariance matrix for class i.
Linear discriminant analysis adds the homoscedasticity assumption, i.e. covariance matrices are identical $\varepsilon_0=\varepsilon_1$, which simplifies the computation and yields a linear decision boundary function. If the covariance matrices are not identical, the surface boundary is a parabola and thus the resulting model is the quadratic discriminant analysis (QDA).

Logistic regression classifier

Logistic regression is a parametric probabilistic discriminative model relying on the sigmoid or logistic function *sigm* which maps real-valued numbers into the unit interval. As a discriminative model, it learns predictive probabilities of class labels given an example only. Denote $\mathbf{x}$ an input vector of d dimensions and $\boldsymbol{\theta}$ a set of real parameters concatenated as a vector. A logistic regression maps a linear combination of the input entries into a non linear output which is the predictive probability:

$$p(Y=y|\mathbf{x}) = sigm(\boldsymbol{\theta}^t\cdot\mathbf{x}) = \frac{1}{1+e^{-\boldsymbol{\theta}^t\cdot\mathbf{x}}}. \tag{1.16}$$

The set of parameters is learned from the training data $\mathscr{D}_{\text{train}}$ using maximum likelihood estimation which is formally equivalent to minimizing a logarithmic cost function $J(\boldsymbol{\theta})$. The

usage of the mean-squared error to learn logistic regression (as in linear regression) yields to a non convex cost function and thus increases the chance of getting stuck in a local optimum. In a binary classification problem, the expression of $J(\boldsymbol{\theta})$ is given by:

$$J(\boldsymbol{\theta}) = \frac{1}{n_{\text{train}}} \sum_{i=1}^{n_{\text{train}}} (-y^{(i)} \log(p(Y = y^{(i)}|\mathbf{x}^{(i)})) - (1 - y^{(i)}) \log(1 - p(Y = y^{(i)}|\mathbf{x}^{(i)}))). \quad (1.17)$$

where n_{train} is the number of training samples, $\mathbf{x}^{(i)}$ is the i^{th} training sample and $y^{(i)}$ is the corresponding label. This cost function is the negative log-likelihood when cross entropy is employed as loss function between the estimated probabilities and the true class labels. Since the objective function is convex, a gradient descent is then used to find the parameters of the model $\hat{\boldsymbol{\theta}} = \min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$. The update of the parameters can be done once for all training samples or sequentially using mini-batches of training data.

Neural networks

Neural networks is a large family of machine learning algorithms that can be used for supervised tasks. The basic building block of a neural network is a neuron which is a computational unit. Each neuron in a neural network is pre-activated by a linear function and then mapped by a (usually) non-linear activation function such as the logistic function *sigm*:

$$\text{Pre}(\mathbf{x}) = b + \mathbf{w}^t \cdot \mathbf{x} \quad \text{: Preactivation function} \quad (1.18)$$

$$\text{Act}(\mathbf{x}) = sigm(b + \mathbf{w}^t \cdot \mathbf{x}) \quad \text{: Activation function} \quad (1.19)$$

where $\mathbf{w}$ is a weight vector and b is a scalar (called bias or intercept). Since the discriminative power of a single neuron is limited, multilayer neural networks were proposed to increase the classification capacity.

A multilayer neural network consists of an input layer, an output layer which corresponds to class label probabilities and at least one intermediate hidden layer. The computation operations inside each neuron remain the same as in equations 1.18 and 1.19. In a fully connected neural net with L layers, there are $L-1$ weight matrices and $L-1$ intercept vectors gathering the parameters connecting a layer vector of activations with its input vector. Each weight matrix $\mathbf{W}$ contains of a set of weight vectors $\mathbf{w}$ and an intercept vector $\mathbf{b}$ contains a set of intercepts values.

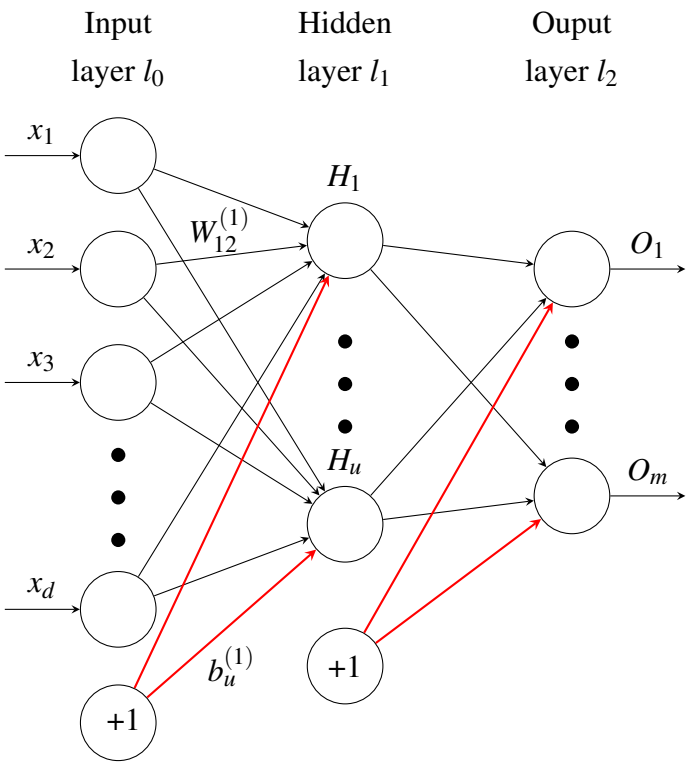


Fig. 1.4 A neural network with 3 layers $\{l_0, l_1, l_2\}$. The hidden layer l_1 contains u units, which is a hyperparameter of the model. The sizes of $\mathbf{W}^{(1)}$ and $\mathbf{W}^{(2)}$ is $u \times d$ and $m \times u$ respectively. (Figure originally coded in Latex by Mark Wibrow and slightly modified [Mar])

Figure 1.4 shows a simple neural network architecture in which there is only one hidden layer. The activations of the hidden and the output layers are given by the vectors $\mathbf{H}$ and $\mathbf{O}$ respectively. For both layers l_1 and l_2, we associate two bias vectors $\mathbf{b}^{(1)}$ and $\mathbf{b}^{(2)}$.
The hidden layer can be expressed by:

$$\mathbf{H} = sigm(\mathbf{b}^{(1)} + \mathbf{W}^{(1)} \cdot \mathbf{x}). \tag{1.20}$$

The output layer is computed via a *softmax* function that produces a vector of posterior probabilities $p(Y = y|\mathbf{x})$ of size m:

$$\mathbf{O} = \left(\frac{e^{Z_1}}{\sum_{j=1}^{m} e^{Z_j}} \dots \frac{e^{Z_m}}{\sum_{j=1}^{m} e^{Z_j}} \right) \text{ with } Z_j = b_j^{(2)} + \mathbf{w}_{j.}^{(2)} \cdot \mathbf{H}, \tag{1.21}$$

where $\mathbf{w}_{j.}^{(2)}$ is the jth line vector in $\mathbf{W}^{(2)}$. As in other probabilistic models, $\hat{c}(\mathbf{x}) = \arg\max_{y \in \Omega} O_y$.
Training a neural network consists in optimizing the set of parameters $\boldsymbol{\theta} = \{\mathbf{W}^{(1)}, \mathbf{b}^{(1)}, \mathbf{W}^{(2)}, \mathbf{b}^{(1)}\}$.

The objective function is essentially the same as in logistic regression. In general, it writes

$$Q(\boldsymbol{\theta}) = \sum_{i=1}^{n_{\text{train}}} \text{los}(\hat{c}(\mathbf{x}), y) + \alpha \text{Reg}(\boldsymbol{\theta}) \tag{1.22}$$

where los is the cross entropy loss function and Reg is a regularizer that penalizes some values of $\boldsymbol{\theta}$ and α is a hyperparameter to regulate the strength of the regularization. Thus, we are looking for $\boldsymbol{\theta}_{\text{opt}} = \arg\min_{\boldsymbol{\theta}} Q(\boldsymbol{\theta})$.

Training a neural net is done using the gradient decent optimization techniques which updates the weights using all training examples. There are other versions such as batch gradient decent where we update the weights using subsets of training examples and stochastic gradient decent that updates the weights for every training sample. Since neural nets have multiple layers, there are many compositions of activation functions hidden in the probabilities $p(Y = y|\mathbf{H})$ and there is no closed form expressions for the gradient of Q with respect to each parameter. Fortunately, the famous backpropagation algorithm allows to perform the gradient update of each parameter in a numerically efficient way starting with parameters of the last layer and progressively updating the parameters of each preceding layer.

1.2.3 Evaluation of classification performances

A principled way to evaluate classifier performances is to derive their frequentist probabilities of success on a validation set $\mathscr{D}_{\text{val}}$. A validation set also contains pairs $\left(\mathbf{x}^{(i)}, y^{(i)}\right)$ that are not exploited during the training phase: $\mathscr{D}_{\text{val}} \cap \mathscr{D}_{\text{train}} = \emptyset$. The size of the validation set is denoted by n_{val}. Of course, practitioners are given a limited number of labeled examples and therefore choosing a large validation set is at the expense of the training set. When the estimated performances are not meant to be used to tune hyperparameters or a combination method (as in our case), a test set is used in place of a validation set. The test set is disjoint from both the validation and train sets. The performances computed from the test set are only meant to compare classification methods within a given benchmark. In the sequel, we present a number of performance criteria computed from a validation set.

The probabilities derived for a classifier from the validation set are estimations of the conditional probabilities $p(\hat{c} = c_j|Y = c_i)$. All these probabilities are derived from the

confusion matrix $\mathbf{M}$ of the classifier:

$$\mathbf{M} = \begin{bmatrix} \sum_{i=1}^{n_{\text{val}}} \mathbb{I}_{c_1}\left(y^{(i)}\right)\mathbb{I}_{c_1}\left(\hat{c}\left(\mathbf{x}^{(i)}\right)\right) & \dots & \sum_{i=1}^{n_{\text{val}}} \mathbb{I}_{c_1}\left(y^{(i)}\right)\mathbb{I}_{c_m}\left(\hat{c}\left(\mathbf{x}^{(i)}\right)\right) \\ \vdots & \ddots & \vdots \\ \sum_{i=1}^{n_{\text{val}}} \mathbb{I}_{c_m}\left(y^{(i)}\right)\mathbb{I}_{c_1}\left(\hat{c}\left(\mathbf{x}^{(i)}\right)\right) & \dots & \sum_{i=1}^{n_{\text{val}}} \mathbb{I}_{c_m}\left(y^{(i)}\right)\mathbb{I}_{c_m}\left(\hat{c}\left(\mathbf{x}^{(i)}\right)\right) \end{bmatrix},$$

where $\mathbb{I}_{c_i}$ is the indicator function for class c_i. The rows and columns refer to the actual and predicted classes respectively. The element M_{ij} is the number of examples whose true label is c_i but were classified as c_j by the classifier. The number $n_{\text{val},i}$ of examples of class c_i in the validation set is given by the sum of elements in the i^{th} row. Normalizing each row by these numbers yields the estimated probabilities

$$\hat{p}(\hat{c} = c_j | Y = c_i) = \frac{\mathrm{M}_{ij}}{\sum_{j=1}^{m} \mathrm{M}_{ij}} = \frac{\mathrm{M}_{ij}}{n_{\text{val},i}}. \tag{1.23}$$

Standard classification performance criteria are also computable from $\mathbf{M}^{(k)}$. *Precision* is defined as:

$$\mathrm{Prec}_i = \frac{\mathrm{M}_{ii}}{\sum_{j=1}^{m} \mathrm{M}_{ij}} = \frac{\mathrm{M}_{ii}}{n_{\text{val},i}}. \tag{1.24}$$

The precision rate of class c_i is interpreted as the probability that a randomly selected example whose predicted class is c_i has a true label c_i.
Similarly the *recall* of class c_j is the probability that a randomly selected example whose true label is c_j, is predicted as c_j:

$$\mathrm{Rec}_j = \frac{\mathrm{M}_{jj}}{\sum_{i=1}^{m} \mathrm{M}_{ij}}. \tag{1.25}$$

Finally, the global *accuracy* of the classifier is given by

$$\mathrm{Acc} = \frac{\sum_{i=1}^{m} \mathrm{M}_{ii}}{\sum_{i=1}^{m}\sum_{j=1}^{m} \mathrm{M}_{ij}} = \frac{\sum_{i=1}^{m} \mathrm{M}_{ii}}{n_{\text{val}}}. \tag{1.26}$$

Receiver operating characteristic (ROC) curves are also an important model evaluation tool. A ROC curve is a graphical plot that illustrates the performance of a binary classifier and can be used to compare a set of classifiers. Before explaining the ROC curve we need to clarify the notions of specificity and sensitivity in machine learning classification tasks. Considering a binary classification problem $\Omega = \{c_1, c_2\}$, the sensitivity and specificity are

the recall values for class c_1 and c_2 respectively:

$$\text{Sensi} = \frac{\text{M}_{11}}{\text{M}_{11} + \text{M}_{21}} \tag{1.27}$$

$$\text{Speci} = \frac{\text{M}_{22}}{\text{M}_{12} + \text{M}_{22}}. \tag{1.28}$$

The ROC curve shows the plot of (1−speci) on the horizontal axis against the sensitivity on the vertical axis and thus we need a set of (1−speci, sensi) points. When constructing the confusion matrix, we chose a regular cut-off threshold probability[2] of 0.5, this means that a sample is assigned to c_i if $p(Y = c_i|\mathbf{x}) > 0.5$ and thus we obtain one pair (1-specificity, sensitivity). When the cost of misclassifying examples from class c_1 is not the same as the cost of misclassifying an example of class c_2, other threshold values are used and we obtain a set of points allowing to construct the ROC curve.

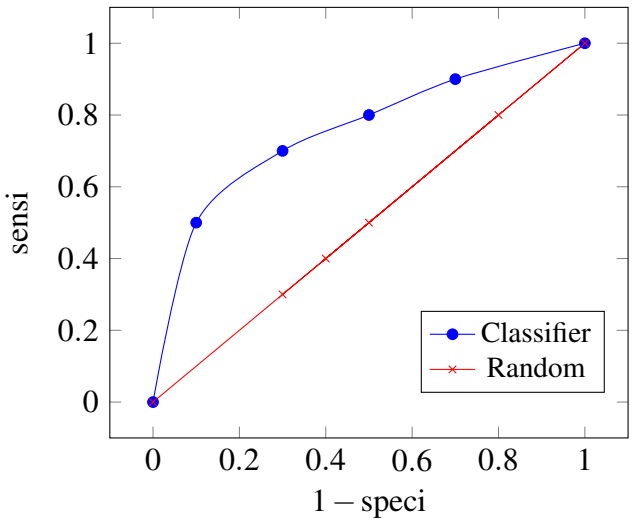


Fig. 1.5 An example of ROC curve.

Examples of ROC curve are given for a random and trained classifier in figure 1.5. The diagonal red line corresponds to the random classifier. Another measure that can be extracted from this plot is the Area under the ROC curve (AUC). Suppose we focus on the score s (e.g. the estimated probability in a logistic regression) returned by the classifier for the class c_2. The AUC is then the probability that a randomly chosen example from class c_2 obtains a value of s greater than the one obtained by a randomly chosen example from class c_1. In the

[2] This threshold is the one corresponding to the classification rule relying on $\arg\max_{y\in\Omega} p(Y = y|\mathbf{x})$ for probabilistic classifiers.

ideal case, we would like to have the blue curve as far as possible from the red line whose AUC is 1/2.

Another interesting classifier evaluation measure is the F-score which combines the precision and the recall measures by computing their harmonic average:

$$\text{F-score} = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}}. \tag{1.29}$$

The previous measures evaluating classifier performances are very important indicators on classifier decision reliability. It may be useful for a user to take them into consideration when combining classifiers because they somehow reflect a degree of confidence on a classifier decision regarding a test sample. Precision, recall, accuracy, AUC and F-score are all types of the contextual information issued from confusion matrices of a given classifier. In this seminal work, we are particularly interested in classifiers combination using contextual information represented by entire confusion matrices since they are more informative than point valued criteria.

1.3 Why do we combine classifiers?

In machine learning, the most important aspect of each classification algorithm is its ability to generalize which means its ability to correctly label an unseen test sample using acquired knowledge from training data. From the no free lunch theorem [99], we know that there is no universally optimal classification algorithm and consequently there is a vast literature on supervised learning techniques. When practitioners have to go for a given algorithm or model for their data and when they try multiple classifiers on a given problem, an idea comes in mind: *why not combining the decisions of those classifiers to get better generalization performances ?*

Based of different insights from the state of the art, we discuss in this section the above question and we try to give some motivations underlying classifier combination and shed light on the circumstances in which it has a significant added value. Many works were dedicated in the last two decades to classifiers fusion under different circumstances and have been experimentally and theoretically demonstrated the efficiency of such methods [7, 36, 8, 9, 101, 78, 84].

We discuss first a number of aspects argued by Dietterich [24] in the particular context of Ensemble of Classifiers (EoC), which consists in combining homogeneous classifiers i.e. classifiers that are instances of the same generic model. According to him, there are several

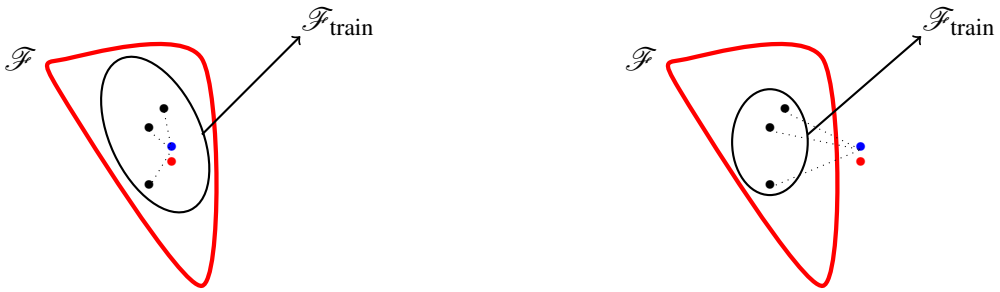
causes accounting for a classifier failure. These issues can be categorized in three types of limitations: statistical, representational and computational.

Bias-Variance tradeoff

- **Statistical limitation:** Supposing a space $\mathscr{F}$ containing all possible decision functions that can be learned by a given model. One may see the learning algorithm as a procedure selecting the best function $f_{\text{opt}} \in \mathscr{F}$ having the maximal generalization ability. This statistical limitation arises when having a small dataset because if we drew several datasets of the same size, each of them would converge to quite different decision functions. In other words, there is large variance in the decision function selection process. However, creating an ensemble of classifiers denoted by $\mathscr{F}_{\text{train}}$ and aggregating their decisions by a voting approach or averaging may decrease the risk of choosing a poor classifier.

- **Representational limitation:** The (expected) distance between the learned function and the optimal one is large because the model is too simple to learn near optimal functions. In other words, the ensemble $\mathscr{F}$ does not contain f_{opt} but fortunately we can extend[3] $\mathscr{F}$ by combining classifiers in $\mathscr{F}_{\text{train}}$. The representational limitation is equivalent to a *bias*.

The Bias-variance tradeoff is a common difficulty in machine learning since usually a low bias goes with a high variance and a low variance goes with a high bias. The two types of limitations are illustrated in figure 1.6.



(a) Statistical limitation (b) Representational limitation

Fig. 1.6 Illustration of the statistical (left) and the representational (right) limitations . Black dots (•) represent learnt decision functions, the blue dot (•) corresponds to the combination of elements in $\mathscr{F}_{\text{train}}$ and the red dot (•) represents f_{opt}. (figure inspired from [24])

[3] The capacity of an ensemble can be greater than the capacity of the base classifiers.

Computational limitation

Beside the variance-bias tradeoff, some classification algorithms face computational issues. This is due to the fact that many of them get stuck in local optima when looking for the optimal decision function f_{opt} in $\mathscr{F}$. Dietterich [24] gives examples of neural nets or decision trees whose optimal training is NP-hard. However, EoC offers a solution for this problem since the starting point for the research of f_{opt} can change with every classifier and thus the risks to get stuck in a local optima decreases. From multiple starting points, one is able to cover many regions in $\mathscr{F}$ and thus circumvent the local optima problem.

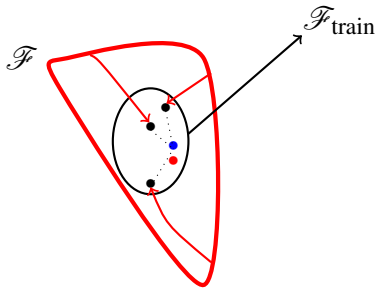


Fig. 1.7 Computational limitation. (figure inspired from [24])

Other motivations

The previous motivations concerning statistical, representational and computational limitations are essentially oriented for homogeneous classifier combination settings in which each base classifier has the same capacity. When combining heterogeneous classifiers, the base classifiers may have very different capacities. Some may overfit while other may underfit. In this case, combination can act as a soft model selection procedure by identifying a subset of models that are neither overfitting nor underfitting.

A user who is working for the first time with the machine learning library `Scikit-learn` in Python will face dozens of algorithms that can be applied, and experience difficulties to choose one without prior knowledge. Actually, this is an important advantage of multiple classifier systems because we may hope to construct a methodology involving less human expertise in the choice of classification algorithms and achieve satisfying performances.

In machine learning real-life applications, a classification task might be very complex (e.g. identifying objects in images : there are virtually an infinity of objects and there may be several objects in an image). If the problem is too complex, it makes sense to break it in simpler classification problems and solve them separately, then combine solutions e.g. train

classifier to recognize some specific classes (dog, train, bike, chair, ..) or subsets of classes (animals, vehicles, furniture, etc.).

1.4 Insights on classifier combination

In this section, we present some generalities to bear in mind when combining classifiers. Specifically, we investigate the taxonomy of the fusion scheme concerning the nature of the combined information, the combination rule, the types of individual classifiers and the structure of the combination. Then, we discuss the importance of diversity in a multi-classifier system and review some methods from the literature to induce it.

1.4.1 Specification of a classifier combination

Four questions are helpful to determine the perspective from which we are attacking the combination problem:

- Q1: What **nature of output information** is to be combined ?
- Q2: What **types of classifiers** are to be combined ?
- Q3: What **combination rule** is used to combine decisions ?
- Q4: What is the **structure** of this combination ?

If these questions are answered, it becomes easier to identify state-of-the-art methods to choose from or to compete with. The possible answers to (Q1) are presented in the introduction of this chapter.

Fusion methods lying on classifiers deriving from the same classification algorithm are called homogeneous combination approaches (e.g. random forest uses decision trees) or ensemble methods.
A standard approach in this category is bagging (**b**ootstrap **agg**regat**ing**) introduced for the first time by Breiman in [7] and developed later in [9, 8]. Another very popular ensemble method is boosting [85] whose most widely used implementation is AdaBoost [36]. Bagging and boosting were well explored and examined by the machine learning community in the last two decades. Recently, a new boosting design that makes use of linear programming support vector machines (LPSVM) was introduced in [79].

If the fusion panel contains different classifiers, then we talk about heterogeneous combination approaches i.e. combining a neural network and a decision tree. These two types of panels (homogeneous and heterogeneous) are the main categories of multiple classifier systems arising from (Q2) .

Let alone the types of combined classifiers, combination methods are also categorized with respect to the rule allowing decision fusion. The two main categories are deterministic [67, 72] versus probabilistic approaches [17, 101, 57, 70] which are two possible answers to (Q3) . By probabilistic approaches, we mean that the classification function relies on the maximization of the class probability distribution. Note that in [72], Lecué uses a convex combination to aggregate binary classifiers therefore this contribution belongs to the deterministic approach category. However, the author uses probabilistic arguments to prove that the investigated combinations minimize the excess hinge risk with optimal convergence rates.

Classifier combination can also be performed within other uncertainty frameworks which may be alternatives to probability theory such as fuzzy logic [14] or related to probability theory such as the belief function framework [81]. In the original paper of Dempster, belief functions are obtained by plugging a probability measure with a multi-valued mapping. Other authors advocate that the theory is self-contained and need not to be built upon probabilistic objects.

Concerning combination architecture (Q4), we mean how classifiers are organized as a network and connected to the fusion process. There are three main categories of topology: parallel, sequential or a hybrid combinations. In parallel fusion, the base classifiers work independently (Figure 1.8), they may be trained with inputs living in different feature spaces and the feature vectors may or may not be derived from the same raw training examples. However, the output of a given classifier cannot serve as input for another classifier. See [67],[47],[101] for reviews containing illustrative examples of parallel classifier fusion techniques. In sequential (serial) fusion, elementary classifiers are stacked in a sequential way and the decision of one classifier depends on a previous decision. Some class labels are eliminated at each classification step until one class is left. Kittler *et al.* addressed this problem in [59] by giving theoretical formulations of such combinations usually organized into coarse to fine classifications as shown in fig 1.9. Hybrid hierarchical fusion consists in a mix of parallel and sequential architectures (see fig 1.10). In [58], hybrid hierarchical fusion was used for the recognition of handwritten words on bank checks and achieves lower error rates.

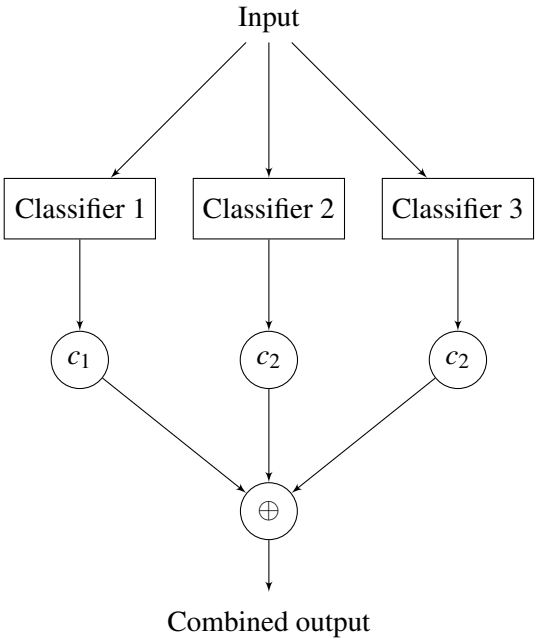


Fig. 1.8 Parallel combination of three classifiers. The predicted classes delivered by the classifiers are the inputs of the fusion operator.

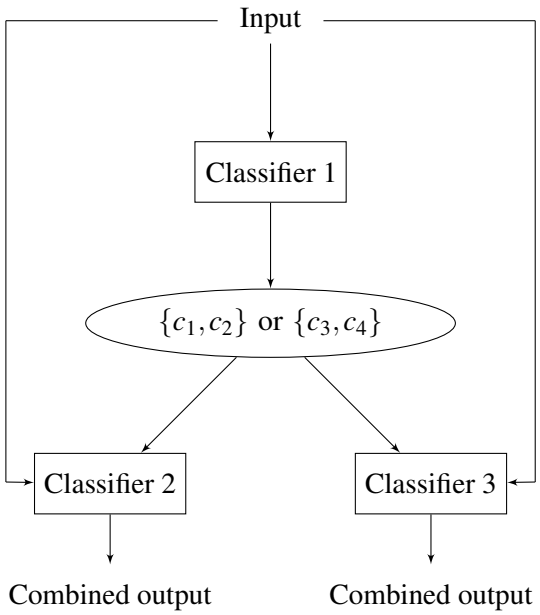


Fig. 1.9 Serial combination of three classifiers. The first classifier assigns the input to a set of candidate classes then the second classifier refines the classification to a single class using both the first classifier output and the input. The response of the first classifier can be viewed as a multiplexer.

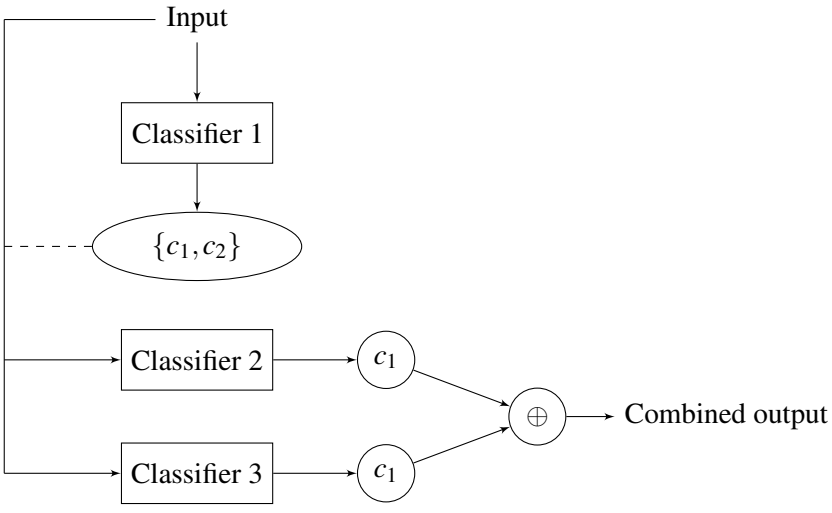


Fig. 1.10 Hybrid hierarchical combination of three classifiers. Classifiers 2 and 3 are built upon input and classifier 1 in a serial fashion. The outputs of classifiers 2 and 3 are combined using a fusion operation in a parallel fashion.

1.4.2 The need of diversity in responses

Designing a multiple classifier system (MCS) requires special care in the choice of individual classifiers so as to achieve higher classification accuracy or obtain more robust algorithms. For instance, surely a panel of linear classifiers trained on the same dataset will converge to very close separating hyperplanes and combining very similar decision rules is doomed to achieve average results. Thus, diversity is a guiding rule to design an efficient multiple classifier system.

The notions of diversity and independence are closely related but the separating line in between is still unclear. However, we might not expect that a highly dependent classifier pool outputs very diverse responses since dependent classifiers will certainly produce correlated responses. In this section, we discuss the ways of measuring diversity as well as the methods to induce diversity in classifier responses.

Brow *et al.* [10] advocates that inducing diversity in a classifier ensemble can be done in two ways. The first way is explicit and it is based on the optimization of the diversity metric over a pool of classifiers. The second one is implicit and it consists in generating classifiers using different mechanisms and hoping to have significant independence in their responses. This is usually done by training them on different learning data samples or on different region of the features space.

Diversity measures

In [100], the authors divided diversity metrics into pairwise and non-pairwise metrics. Using pairwise diversity measures, we build an MCS by optimizing a diversity measure averaged on each pair in the classifier ensemble. Examples of such measures are kappa-statistics [76] or Q-statistic [10]. Other types of measure can be also employed to build the multiple classifier system such as double-fault measure [69, 41] which is the proportion of test samples for which two classifiers give both incorrect answers. Disagreement [89] is defined as the proportion of test samples for which two classifiers produce different outputs such that one of them responds correctly. A non-pairwise diversity measure has a *one versus all* scheme therefore it compares a classifier output to all other classifiers in the system. In this family, we can cite the entropy measure used by Cunningham and Carney [16] and the coincident failure diversity [88]. More details about different types of diversity measures can be found in [44],[62],[34] and [100].

Implicit diversity induction schemes

The second category of diversity induction involves some implicit approaches based on the manipulation of either the input data, the output decisions or the types of classifiers.

In their survey [100], Wozniak *et al.* formulated the following taxonomy for data manipulation:

- *Partitioning the data points*: this allows to train individual classifiers on different training sets. Even though learning samples are somehow correlated, we keep a certain level of diversity in classifier responses using different training data that might be obtained by a split or by a random sampling of learning samples as in Bagging [7–9]. In some experiments in chapter 3, we split the dataset so that, for each subset of data sharing the same class label, each split belongs to non overlapping regions of the feature space. In this case, the base classifiers have the ability to specialize themselves and are thus a lot more diverse than in the bootstrap sampling scheme used in bagging.

- *Selecting subsets of features*: again, base classifiers are thus trained on different datasets in the sense that they have access to different pieces of information for each training example. Thus, decision functions have different domains and if the features are weakly correlated then we achieve diversity. A popular strategy in this vein is to randomly select the subset of those features selected to train each base classifier. This strategy is known as the random subspaces method [46]. Another strategy consists in selecting the maximally accurate individual classifier for each partition in the feature space [5, 50, 61]. In this setting, features are not randomly sampled but clustered using some existing algorithms such as Kuncheva's algorithm [65] and the adaptive splitting proposed in [53].

- *Modifying classifier outputs*: MCS can enjoy another form of diversity by asking each classifier to discriminate only a subset of classes.To recover a decision granularity of the whole set of classes, it is necessary to design a specific information fusion strategy. A famous approach in this category are *one-vs-all* classifiers in which case the problem of multi-class classification problem is decomposed into binary classification problems. The classifier will choose between a given label and the remaining labels. The error-correcting output codes method [25] also belongs to this family of approaches. In this setting, each label has a codeword (sequence of binary values) and then binary classifiers are built to produce a *codeword* for each test sample. The chosen label is the one having the closest hamming distance to the test sample codeword.

Diversity can be easily created for classifier ensembles that are based on different learners. However, individual classifiers can be based on the same classification algorithm. In this case, hyperparameter values are different, i.e. a pool of decision trees with different maximum depth, or a pool of neural nets with different weight initializations. The pool can be also constructed with heterogeneous classifiers i.e. combining a neural network, a decision tree and a support vector machine. But to obtain diversity one should select training algorithms relying on pretty different hypotheses sets from which they select their decision functions.

Intuitively, we induce diversity because combining very correlated outputs may be as informative as using a single classifier and thus the fusion process is not justified. Note that creating diversity leads to a decrease in the dependence between classifiers but it does not wipe it out completely. Therefore, in addition to our comment in section 1.2.3 explaining that the combination algorithms proposed in this thesis utilize contextual information represented by confusion matrices to achieve better classification performances, the second original idea of these seminal works is to design combination algorithms accounting for dependences in classifier outputs. In particular, we propose in the following chapters two parameterized combination techniques where the parameter captures the level of dependence between classifiers.

1.5 Classical combination algorithms

In the sequel of this chapter, we review combination algorithms starting with some of the earliest methods introduced in this field. We introduce in the following subsections some variants of voting and Borda counts fusion methods. We also present the Behavior Knowledge Space (BKS) fusion algorithm.

In the remainder of this manuscript, we denote by $\hat{c}_k(\mathbf{x})$ the decision of the k^{th} classifier about the test sample $\mathbf{x}$ and by K the number of classifiers to be combined.

1.5.1 Voting approaches

An obvious way to reconcile a pool of decisions is to resort to a voting system. Several variants of voting schemes can be employed for classifier fusion and some of them can integrate contextual information such as individual classifier accuracies. The idea behind the voting principle is based on the intuition that the higher the number of classifiers is, the more likely to have a correct final decision. Popular voting schemes are the following ones:

- *Unanimity voting* is a system in which the combined decision is the label c_i if all individual classifier decisions are c_i , otherwise the combined classifier rejects the input $\mathbf{x}$.

$$\hat{c}(\mathbf{x}) = c_i \iff \hat{c}_k(\mathbf{x}) = c_i \qquad \forall k \in \{1,..,K\} \tag{1.30}$$

- *Modified unanimity voting* is a system in which the combined decision is the label c_i if all individual classifier decisions are either c_i or c_{rej}, otherwise the combined classifier rejects the input $\mathbf{x}$. In other words, no classifier should predict $c_j \neq c_i$.

$$\hat{c}(\mathbf{x}) = c_i \iff \hat{c}_k(\mathbf{x}) \in \{c_{\text{rej}}, c_i\} \qquad \forall k \in \{1,\ldots,K\} \tag{1.31}$$

- *Majority voting* is a system in which the combined decision is the label c_i if the number of classifiers predicting c_i is maximal.

Voting methods were widely explored in the literature. It is found in numerous papers as a reference method to compare with. In [101], voting methods were studied using four elementary classifiers trained on the U.S. Zipcode database. A threshold on the number of votes of the selected final label is considered in the paper. Voting was also tested in [60], on the M2VTS database (speech and videos of humans) and on the CEDAR-CDROM database for handwritten digits recognition found on envelops provided by the US Postal Service, using four classifiers. It led to higher accuracies and outperformed some of the examined probabilistic combination rules applied on the posterior probabilities of the classifiers in this study. Although, fusion voting-based methods are based on a strong and logical intuition they are not very robust in the sense that we need a lot of base classifier to perform well in order to take correct decisions. Matan [78] studied the discrimination ability of majority voting ensembles and defined upper and lower bounds of their classification performances in terms of individual classifier performances.

1.5.2 Borda and *w*Borda counts

Borda counts is a rank-based combination scheme where each classifier ranks the classes (candidates) according to their chances to be the correct (true) class. Each rank associated to a score starting from $m-1$ for the first rank to 0 for the last one where m is the total number of classes. In a second time, the sum of the scores that each class has received is calculated, and the class label achieving highest accumulated score is the ensemble prediction. There may be ties, i.e. several class labels maximizing the accumulated score in which case, an

additional algorithmic step is necessary to resolve the tie. A baseline strategy consists in randomly picking one of the label with maximal cumulative score. The Borda method was developed by a French politician named Jean-Charles de Borda in the quest for a genuinely democratic electoral system. It is used currently to elect the deputies in the Republic de Nauru and the Republic of Slovenia.

Example 2. Suppose there are four candidate class labels $\Omega = \{c_1, c_2, c_3, c_4\}$. For three classifiers whose decision functions are ($\hat{c}_1, \hat{c}_2$ and $\hat{c}_3$) Borda counts works as follows: Each classifier $\hat{c}_k$ ranks the class labels (Table 1.1) and then the sum of accumulated scores of each label is calculated (Table 1.2)

Classifiers	Labels ranking
$\hat{c}_1$	$c_2 - c_3 - c_1 - c_4$
$\hat{c}_2$	$c_3 - c_1 - c_2 - c_4$
$\hat{c}_3$	$c_3 - c_2 - c_1 - c_4$

Table 1.1 Ranks of the four candidates (labels)

Candidate	Score
c_1	$1+2+1=4$
c_2	$3+1+2=6$
c_3	$3+3+2=8$
c_4	$0+0+0=0$

Table 1.2 Candidates and their accumulated points

Borda counts is considered to be one of the simplest non-linear combination algorithm. In some cases (mostly when the number of classifiers is large) a conflict can be observed between the opinion of the majority and the decision deduced using Borda's method. That issue was evoked by Parker in [80]. In Borda counts *the difference in scores for every two consecutive candidates ranked by a single classifier is one* so that all the candidates are "*equally distant*" and distributed in a uniform manner on the same axis but this hypothesis is not always justified. For instance, a probabilistic classifier may assign very high probabilities to two labels and very low probability to two other labels (e.g. for a three classes classification problem, with probabilities 49%, 48% and 3%; obviously, when applying Borda counts combination method, having a unit in difference between every consecutive labels does not seem to reflect the classifier level of confidence). Parker advocates that it may be beneficent sometimes to ignore the uniformity assumption and weight the ranks instead. Fishburn [33]

argues that contextual knowledge issued from confusion matrices can be used to estimate those weights. Assigning such weights to candidate labels is considered as a generalization of the uniform Borda counts and usually referred to as *w*Borda.

Example 3. Considering m class labels, when uniformity is applied, an item of rank r will be assigned a score equal to $m-r$ but in *w*Borda it can be assigned a score equal to $(m-r)\times w^{(r-1)}$ where w is the weight. The difference between Borda and *w*Borda is illustrated in Figure 1.11:

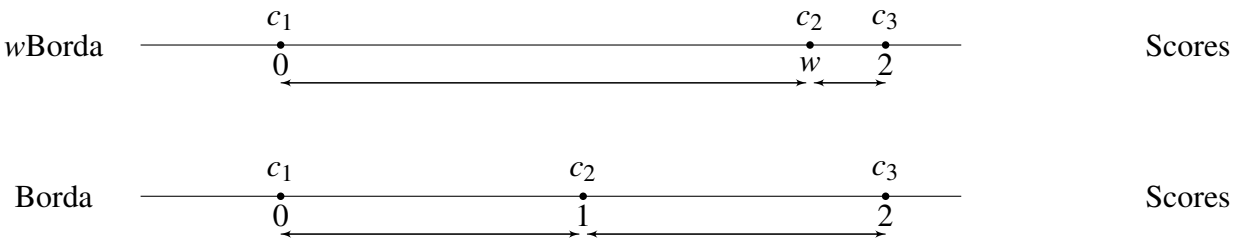


Fig. 1.11 Comparison between Borda and *w*Borda ranking for one classifier which assigns to class label c_1, c_2 and c_3 the rank 3, 2 and 1 respectively. By convention, the class label with rank 1 is the one preferred by the classifier. In regular Borda counts, the scores obtained by c_1, c_2 and c_3 is 0, 1 and 2 respectively but in *w*Borda c_1 will get (3-3)$\times w^2 = 0$, c_2 will get (3-2)$\times w^1 = w$ and c_3 will get (3-1)$\times w^0 = 2$.

1.5.3 Behavior-knowledge space

Huang and Suen [51, 52] proposed the Behavior-Knowledge Space (BKS) method that has the advantage not to rely on prerequisite hypotheses such as statistical independence of classifier outputs.

By definition a BKS is a $K-$dimensional space where the k^{th} dimension is related to the k^{th} classifier. On each dimension k, m decisions can be taken which corresponds to all possible labels in Ω . For each test sample $\mathbf{x}$, we obtain a vector of decisions $(\hat{c}_1(\mathbf{x}),..,\hat{c}_K(\mathbf{x}))$ which corresponds to a point in the BKS. For each point out of the m^K possible ones, we keep track of the number of training samples mapped to this point as well as their class labels.

BKS is built at training time and it has m^K nodes to be learned. Thus, the method does not scale well in either the number of base classifiers or the number of classes as there may be some configurations that are never visited. To decide the class of a test sample, the outputs $(\hat{c}_1(\mathbf{x}),..,\hat{c}_K(\mathbf{x}))$ are obtained and mapped to the corresponding BKS node. The predicted label is the one with the highest number of occurences in this node (Figure.1.12).

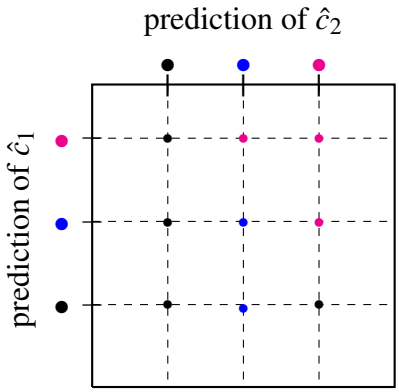


Fig. 1.12 A BKS containing 9 nodes arising from three classes $\Omega = \{\bullet;\bullet;\bullet\}$ and two classifiers. From this BKS, we understand that when both classifier predict $\bullet$, then most of the time, the correct answer is $\bullet$. When $\hat{c}_1$ predict $\bullet$ while $\hat{c}_2$ predicts $\bullet$, then most of the time, the correct answer is $\bullet$.

1.6 Ensemble methods

In this section, we present homogeneous classifier combination methods, i.e. methods in which the base classifiers are instances of the same learning algorithm. A textbook example of homogeneous classifier fusion are random forests which are ensembles of decision trees. In the literature, those methods are often referred to as « ensemble methods » as opposed to heterogeneous combination referred to as « multiple classifier systems ». In this seminal work, we comply with this terminology, however, we use the term « ensemble of classifiers » for either homogeneous or heterogeneous combinations. The distinction between these latter is either clear from context or made explicit. As we mentioned earlier, there is little interest in training classifiers of the same type on the same training set because they will produce the same learning function. Consequently ensemble methods are closely related to diversity induction methods.

In her book entitled «Combining Pattern Classifiers», Kuncheva [67] established a taxonomy for ensemble methods. She considered four possible levels of induction: Data level, Feature level, Classifier level and Combination operator level. Bernard [6] advocates that the fourth level corresponds to the information fusion paradigm and not really to a category of ensemble methods. Let us review the three other levels:

- Data level: This set of methods rely on the construction of several (sub-)datasets on which base classifiers will be separately trained. Such a philosophy was supported by

Xu in [101] to mitigate the independence problem between the base classifiers. Two very well known methods in this category are Bagging [7–9] and Boosting [85, 79].

- Feature level: Each base classifier has access to all training examples but for each example only a predefined subset of the features can be used. Assigning a subset of feature of each base classifier is the most challenging aspect of such ensemble methods. In this level we can cite Ho [46] who introduced the feature bagging method (also known as Random Subspaces) where base classifiers are trained on randomly selected features.

- Classifier level: For a given training algorithm, a simple way to obtain a homogeneous ensemble of classifiers is to use several hyperparameter values to run a training session for each value. For instance, we can create an ensemble of k-nearest neighbors classifiers by using several value of k.

In the rest of this section, we present some well known ensemble methods. In particular, we present the basic versions of Bagging, Boosting , Random subspaces and Error-correcting output codes approaches.

1.6.1 Bagging

The ensemble method referred to as Bagging performs the combination at the data level. Several datasets are obtained by sampling uniformly at random r examples from the original dataset. The probabilities to select a given example in the original dataset does not evolve from a sampling step to another. For a sampled dataset having the same size as the original set ($r = n_{\text{train}}$), it is expected to get approximately 63% of the observations of the original examples (Figure 1.13).

Bagging (**B**ootstrap **Agg**rega**ting**) is based on the non-parametric Bootstrap sampling technique which is a statistical method used to analyze the variability of an estimate and quantify its uncertainty (by computing confidence intervals for instance). Intuitively, if the estimate values are close whenever we use a different dataset (obtained by draws with replacement from the original sample), then we can have a high level of confidence in the estimate. This idea is valid if the empirical distribution is a good approximation of the true population distribution.

After sampling, each classifier is trained on a single bootstrap sample. The decision of all classifiers is often combined using majority voting.

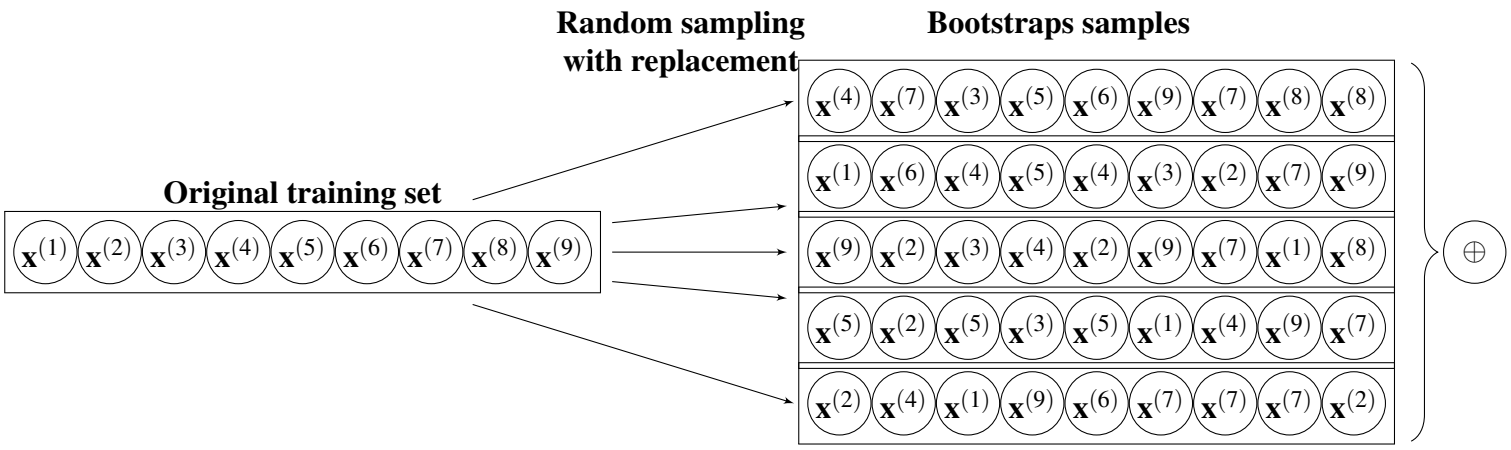


Fig. 1.13 Illustration of the Bagging principle. Each input $\mathbf{x}$ is a training sample.

1.6.2 Boosting

Boosting is a family of algorithms that can be seen as a set of solutions to the question proposed by Kearns's [55]: «*can an ensemble of weak learners create a stronger one?*». A classifier whose discriminative power is slightly stronger than a random one is called a *weak learner*. Thus a trained weak learner $\mathscr{L}$ yields classifiers with accuracy rates that are slightly above $\frac{1}{2}$ in a binary classification problem.

In fact, Boosting is a sequential homogeneous combination method where a set of classifiers, that are iteratively computed in a way that the k^{th} classifier $\hat{c}_k$ focuses on correctly classifying those examples misclassified by the previous classifiers $\hat{c}_{k-1},..,\hat{c}_1$ The base classifiers are linearly combined: $\hat{c}_c = \sum_{k=1}^{K} \beta_k.\hat{c}_k$ where β_k are combination parameters known in closed form (see Algorithm 1). To allow a given base classifier to focus on some training examples, a weight $D_k^{(i)}$ is assigned to each data point $\mathbf{x}^{(i)}$. Weights are also known in closed form. Note that the weak learner $\mathscr{L}$ needs to be able to be run using a weighted loss instead of the usual loss but it is not a major difficulty in general. The algorithm starts with classifier $\hat{c}_1$ that is trained on a uniform distribution of all classes on the original dataset, i.e. all weights are constant one. The weights are then updated based on the errors of prediction in a way to increase the weight of the learning data that has been misclassified by this classifier, while simultaneously decreasing the weights of the well classified data. Thus, we gradually specialize classifiers.

In his famous paper, Schapire [85] proved the surprising equivalence of the notions of *weak* and *strong learnability* which theoretically means that a strong learner can be obtained from weak learners. He then introduced his first recursive boosting algorithm. A more

advanced boosting algorithm was presented later by Schapire and Freund [36]. The first version, AdaBoost.M1 is described in algorithm 1.

Algorithm 1: Adaptive Boosting (AdaBoost)

Data: Weak learner $\mathscr{L}$, $\mathscr{D}_{\text{train}}\{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{n_{\text{train}}}$, K

for $i \in \{1, \ldots, n_{train}\}$ **do**
  $D_1^{(i)} = \frac{1}{n_{\text{train}}}$ /* Uniform initialization of the weights */
for $k \in \{1, \ldots, K\}$ **do**
  Call $\mathscr{L}(\mathscr{D}_{\text{train}}, D_k^{(1)}, \ldots, D_k^{(n_{\text{train}})})$ /* learning $\hat{c}_k$ */
  $\hat{\varepsilon}_k = \sum_{i:\hat{c}_k(\mathbf{x}^{(i)}) \neq y^{(i)}} D_k^{(i)}$ /* Compute the error at the k^{th} iteration */
  if $\hat{\varepsilon}_k > 0.5$ **then**
    Abort loop and discard $\hat{c}_k$
  else
    Compute coefficient $\beta_k = \frac{\hat{\varepsilon}_k}{1-\hat{\varepsilon}_k}$
  for $i \in \{1, \ldots, n_{train}\}$ **do**
    if $\hat{c}_k(\boldsymbol{x}^{(i)}) = y^{(i)}$ **then**
      $D_{k+1}^{(i)} = D_k^{(i)} \times \beta_k$
      /* Update the weights D */
    else
      $D_{k+1}^{(i)} = D_k^{(i)}$
  Renormalize weights so that $\sum_{i=1}^{n_{\text{train}}} D_{k+1}^{(i)} = 1$
Return mapping $\mathbf{x} \rightarrow \arg\max_{y \in \Omega} \sum_{k:\hat{c}_k(\mathbf{x})=y} \log \frac{1}{\beta_k}$ /* Combined classifier */

1.6.3 Random subspaces

The idea of the random subspace method (RSM) also known as feature bagging is quite similar to "ordinary" bagging but instead of sampling uniformly at random examples with replacement, we sample features with the same mechanism [46, 47, 11]. The principle of this method consists in training the classifiers in random subspaces of dimensionality lower than the dimensionality of the original space. Ho [46] proved that the best results are obtained when the original space dimensionality is approximately reduced by a half.

One of the advantages of the random subspaces method lies in its ability to insure stability in classification. Input vectors in high dimensional spaces are notoriously more difficult to discriminate, a phenomenon often referred to as the curse of dimensionality. An example

of RSM is illustrated in Figure 1.14, where 4 features out of 9 are sampled to train three individual classifiers.

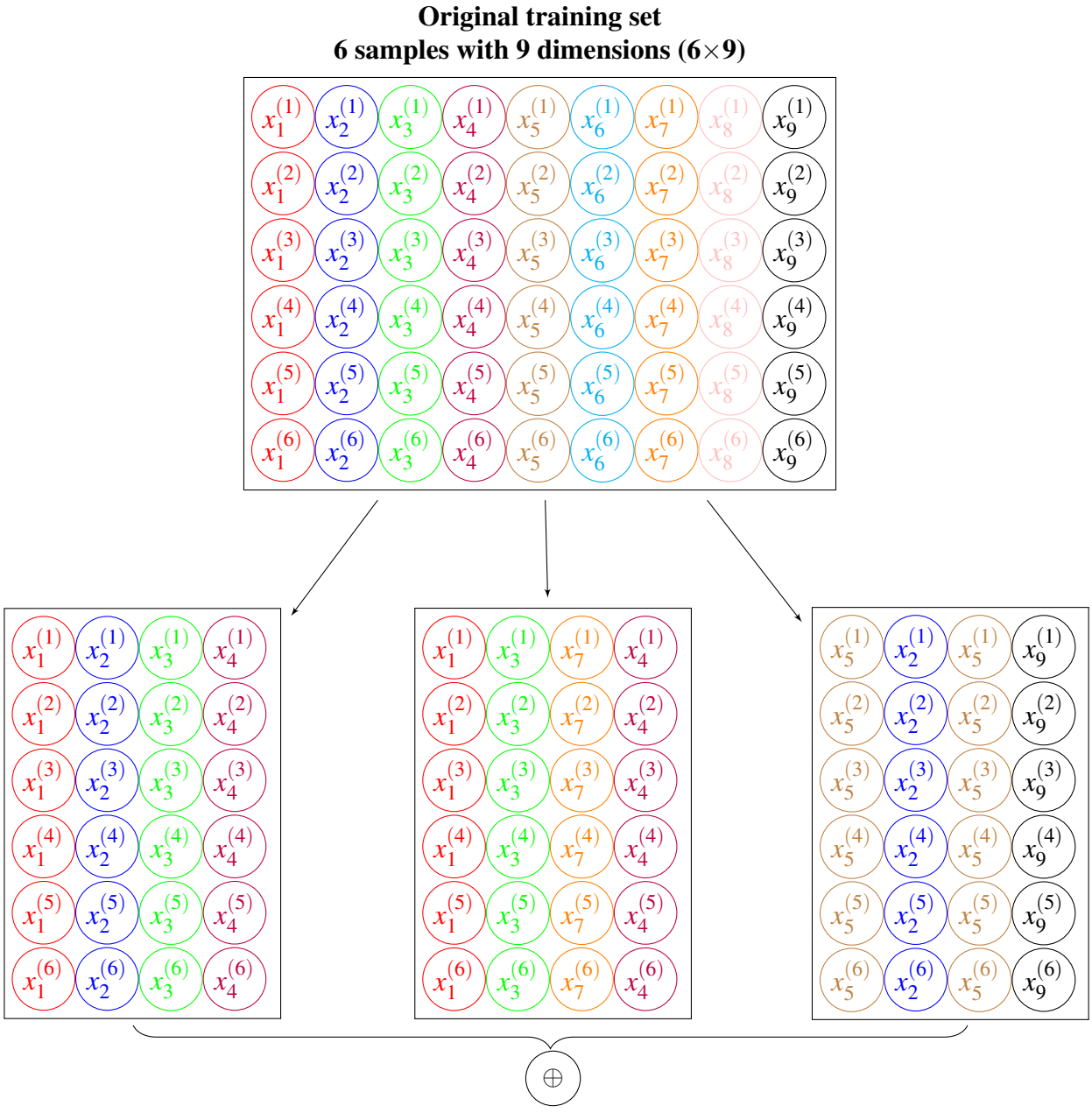


Fig. 1.14 Illustration of the Random subspace principle. Each $\mathbf{x}^{(i)}$ is an input, i.e. a training sample vector with 9 dimensions (9 features). Four features out of nine are sampled for three individual classifiers.

1.6.4 Error correcting output codes

Error-correcting output codes (ECOC) [26] are a meta-classifier algorithm belonging to ensemble methods. It solves multiclass classification problems by breaking them into mini binary classification ones. It consists in representing each class label by a codeword (string of «0» and«1») of length K where K is also the number of classifiers involved in the fusion. Then, each classifier discriminates between two subsets of Ω and outputs a binary value representing a subset of labels. The resulting codeword of a test sample $\mathbf{x}$ must be of length K. To decide the label of $\mathbf{x}$, we choose the label having the closest codeword to the obtained codeword of $\mathbf{x}$ in term of Hamming distance (number of unequal bits).

The number of bits to encode class labels is always greater than necessary. We need at least $\log_2 m$ bits to encode m class labels. If we choose more bits, we can hope for a form of robustness arising from the redundancy of the base classifier decisions.

Example 4. Let $\Omega = \{c_1, c_2, c_3, c_4\}$ denotes the set of class labels ($m = 4$). One can construct the following codeword table using an ensemble of 7 classifiers $\{\hat{c}_1, .., \hat{c}_7\}$.

Labels	$\hat{c}_1$	$\hat{c}_2$	$\hat{c}_3$	$\hat{c}_4$	$\hat{c}_5$	$\hat{c}_6$	$\hat{c}_7$
c_1	1	1	1	1	1	1	1
c_2	0	0	0	0	1	1	1
c_3	0	0	1	1	0	0	1
c_4	0	1	0	1	0	1	0

Table 1.3 Error-correcting output codes for a four-class problem.

Using these arbitrary codewords, each label is encoded by 7 bits. During learning, each classifier $\hat{c}_k$ is trained to discriminate two subsets of classes. For instance, $\hat{c}_5$ learns to discriminate between $\{c_1, c_2\}$ and $\{c_3, c_4\}$. A test example $\mathbf{x}$ may be mapped to a codeword that is not in table 1.3 and we use Hamming distance to decide to which label $\mathbf{x}$ it belongs.

1.7 Trainable fusion approaches

In this section we discuss another category of algorithms that address the classifier combination problem by training a second stage classifier that learns to perform the fusion. In contrast to other methods such as voting,bagging or Borda counts, these algorithms pass through a training phase where the combination rule or the combination parameters are explicitly deduced from data. In this category, we explore in particular stacking approaches, mixture

models and mixture of experts. However, we may find many other combination schemes −with a learning phase− that can fall in this category of methods such as boosting.

1.7.1 Stacking approaches

Stacking [98, 95, 96, 86] is about learning the combination rule from several sets of base classifier outputs. The stacking framework consists of two levels: a base classification level and a meta-classifier level. In the base level, individual classifiers are built on training data from $\mathscr{D}_{\text{train}}$ (as usual). Each trained base classifier classifies examples from a validation set to constitute a new training set whose samples are classifier outputs. In the meta-level, these vectors are used to train the meta-classifier which maps base classifier outputs to the predicted class.

The aim of the first level is to produce a training set to learn the combination rule. Depending on the output type of the base classifiers, the entries of each example of the second stage training set are either class labels, class ranks or class probabilities. Classifier outputs of the first level are concatenated to form second stage input training vectors for the meta classifier regardless of the possible variability in the output types. The meta-classifier has to cope with this rather arbitrary way of embedding classifier outputs into an artificial feature space. The difficulty in stacking combination approaches lies, similarly as in BKS, in the imbalanced aspect of the second stage training set. Alleging classifier outputs are just class labels, there may be configurations that are never visited at training time but occur at test time. To prevent such a situation, it may be tempting to choose a large validation set but this is at the expanse to the training set and the base classifier accuracies.

Ting and Witten [95] concatenated posterior probability distributions resulting from a classifier stacking in the first level to build training input vectors for the meta-classifier. They discussed the usage of multi response linear regression to learn the combination rule and demonstrated that it outperforms other classification techniques [95]. An interesting approach, proposed in [96] by Todorovski *et al.*, employs properties of the posterior distribution, such as entropy, as input feature vectors to learn a meta-decision tree classifier. The authors report that their method outperforms other combination techniques such as bagging and boosting. Dzeroski and Zenko [31] showed in a comparative study that combining heterogeneous classifiers by stacking yields to close performances as compared to classifier selection by cross validation.

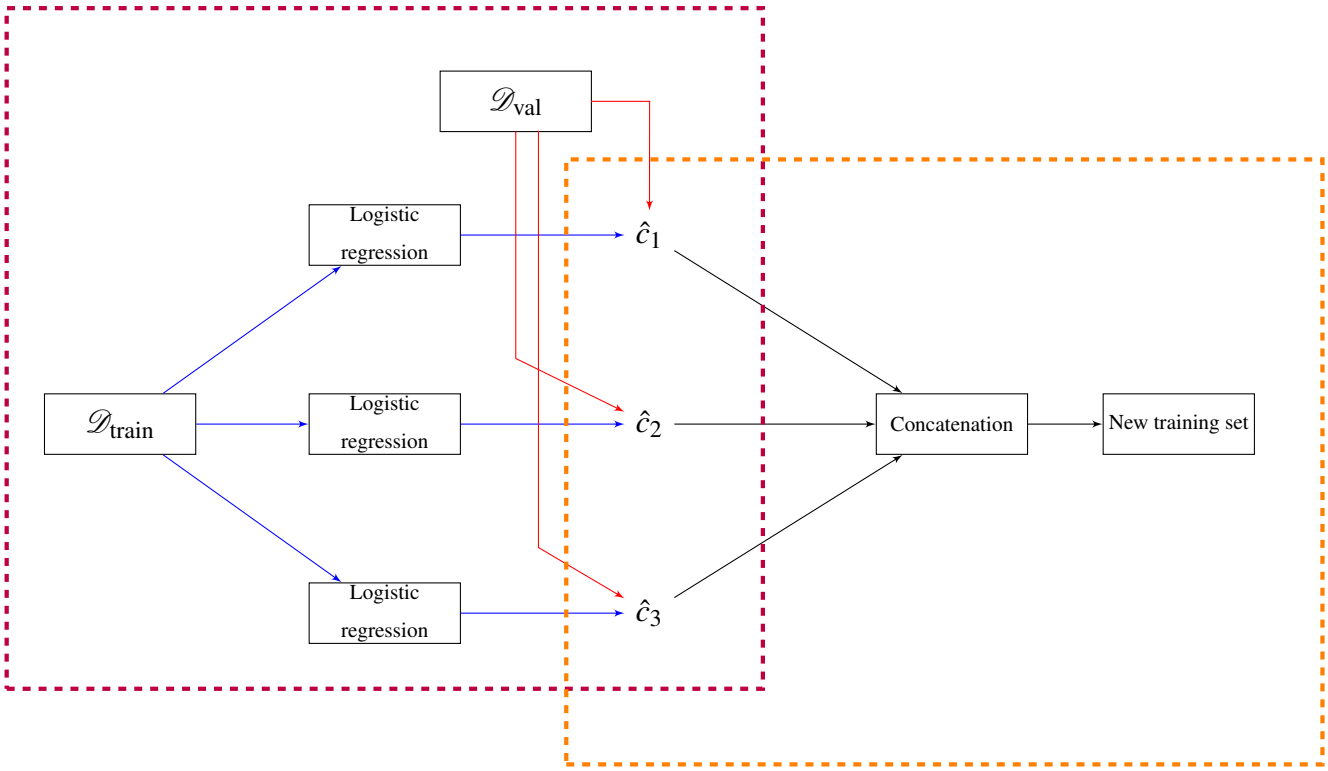


Fig. 1.15 Illustration of logistic regression stacking principle for $K = 3$. The base classifiers are first learned on $\mathscr{D}_{\text{train}}$ to get $\hat{c}_1, \hat{c}_2$ and $\hat{c}_3$. Then they are applied on $\mathscr{D}_{\text{val}}$ and the outputs are concatenated to get a new training set on which a new logistic regression classifier is trained to obtain the combination learning function $\hat{c}_c$.

1.7.2 Mixture models

Mixture models can also be regarded as combination training methods. These models rely on the estimation of the predictive distribution which is alleged to be a mixture of distributions. A mixture model is a convex combination of multiple predictive distributions, each produced by a probabilistic classifier and controlled by a set of parameters $\boldsymbol{\theta_k}$. Suppose we have to combine K classifiers, each producing a predictive distribution $p_{\boldsymbol{\theta_k}}(Y=y|\mathbf{x})$ where k is the index of the k^{th} classifier. The combined predictive distribution reads:

$$p_{\boldsymbol{\Theta}}(Y=y|\mathbf{x}) = \sum_{k=1}^{K} w_k p_{\boldsymbol{\theta_k}}(Y=y|\mathbf{x}) \tag{1.32}$$

where $\{w_1,..,w_K\}$ are the combination parameters ($\sum_{k=1}^{K} w_k = 1$ and $w_k \geq 0, \forall k$) and $\boldsymbol{\Theta} = \{\boldsymbol{\theta_k},..,\boldsymbol{\theta_k},w_1,..,w_K\}$ is the set of all parameters in the mixture. For instance, a mixture model of logistic regression classifiers will have as $\boldsymbol{\theta_k}$ the parameters of the k^{th} logistic regression while a mixture of Gaussians will have as $\boldsymbol{\theta_k}$ the mean and the variance of the normal distribution. Since mixture model demands predictive probabilities thus it is impossible to build them using deterministic classifiers i.e. support vector machines or nearest neighbors. All parameters are jointly estimated from the training set $\mathscr{D}_{\text{train}}$ using Expectation-Maximization algorithm (EM) and we do not take some data out of the training set to build a validation set. EM is a procedure that computes estimates of $\boldsymbol{\Theta}$ by introducing a latent variable Z which has a multinomial distribution and represent the identity of the model component from which a pair $(\mathbf{x},y)$ arises. The EM procedure relies on two macro-steps. The first one, called the *E-step*, consists in finding the expectation[4] of the complete log likelihood function. The likelihood function is expressed by:

$$\mathscr{L}(\boldsymbol{\Theta}) = \prod_{i=1}^{n_{\text{train}}} p(y^{(i)},z^{(i)}|\mathbf{x}^{(i)},\boldsymbol{\Theta}). \tag{1.33}$$

In the second step, the *M-step*, we find the value of $\boldsymbol{\Theta}$ that maximizes the log likelihood using gradient ascent. Even though it insures convergence, the EM procedure does not guarantee convergence to the maximum likelihood estimate. For detailed explanation we recommend the book of Casella and Berger [13].

[4]This expecatation consists in integrating out Z which we do not observe.

1.7.3 Mixture of experts

Mixtures of experts [77] were introduced in the early 90's. Mixtures of experts are quite similar to mixture models in spirit but they differ in the fact that the combination parameters $\{w_k\}_1^K$, also called mixing weights, are input-dependent. In fact, mixtures of experts can be seen as a generalization of mixture models but instead of keeping the same weights , we change them with respect to the input value. In this context, we call the set of mixing weights $\{w_k\}_1^K$ gating functions. Usually, the relation linking mixing weights with inputs is a Softmax function:

$$w_k^i(\mathbf{x}^{(i)}) = \frac{e^{\mathbf{v}_k^t \cdot \mathbf{x}^{(i)}}}{\sum_{k'=1}^{K} e^{\mathbf{v}_{k'}^t \cdot \mathbf{x}^{(i)}}} \tag{1.34}$$

where $\mathbf{V} = [\mathbf{v}_1 \dots \mathbf{v}_K]$ the set of parameters of allowing to compute $w_k^i(\mathbf{x}^{(i)})$ and thus the resulting combined predictive distribution, for a sample $\mathbf{x}^{(i)}$, is given by:

$$p_{\boldsymbol{\Theta}}(Y = y|\mathbf{x}^{(i)}) = \sum_{k=1}^{K} w_k^i(\mathbf{x}^{(i)}).p_{\boldsymbol{\theta_k}}(Y = y|\mathbf{x}^{(i)}) \tag{1.35}$$

The key idea of this setting is to let the gating functions decide what expert should have the greatest impact on the combination for each sample. This allows local specialization of the experts where each of them is specialized in a region of the input space where it has the maximal individual accuracy.
In this approach, the set of the whole model parameters $\boldsymbol{\Theta} = \{\boldsymbol{\theta_1}, .., \boldsymbol{\theta_K}, \mathbf{V}\}$ is learned on $\mathscr{D}_{\text{train}}$ using the EM algorithm as in mixture models.

1.8 Fusion approaches within uncertainty frameworks

This section is dedicated to fusion approaches within an uncertainty framework. In particular, we review some of the existing combination methods in the probabilistic, evidential and fuzzy frameworks. Even though we are interested in combining classifiers based on their estimated performances, the algorithms of this section are not restricted to this setting but they encompass other combination approaches that may be based on different types of scored outputs.

1.8.1 Probabilistic classifier combination

Simple combination algorithms such as majority voting and Borda counts work relatively well in many applications but they suffer from a major limitation which is the absence of any contextual information guiding the fusion toward more reliable solutions. Contextual information can have many forms. A popular type of contextual data are the individual performances of the classifiers. Indeed, given an estimation of the classifier performances, one can try to design a fusion approach so that successful classifiers have a greater impact on the aggregated results than those with limited success. Contextual information can be integrated as part of weighted voting algorithms. Yet it is not always clear how to compute the weights and if this way to integrate contextual data is justified. However, a clever way to benefit from contextual information consists in integrating it in the combination within an uncertainty framework such as the probabilistic framework especially if contextual information is probability distributions. The reason is that the probabilistic framework provides a panel of probabilistic rules permitting to easily manipulate probability distributions.

In this section, we review three probabilistic combination categories of methods. We start with a review of the mainstream approaches in probabilistic fusion, then we introduce two classes of probabilistic pooling methods and we entail this section with some combination techniques within the Bayesian paradigm.

A baseline probabilistic approach

In subsection 1.2.3, we explained that confusion matrices characterize the performances of a classifier. These latter can be exploited to combine classifiers as part of deterministic algorithms. For instance, they were used by Giompapa *et al*. [43] to break ties in majority voting classifier combination applied to naval target classification. In a sequential architecture, Jeong *et al.* [54] select a set of candidate classes based on the confusion matrix of a first level classifier. A second level classifier then evaluates how likely each member of this set is to be the true class. This approach was applied to handwritten Korean character recognition. Taking inspiration from [101], Parker [80] uses confusion matrices to rank class predictions for classifiers producing elementary outputs. A weighted Borda vote is then used to aggregate rankings.

In contrast with the above mentioned approaches, we focus in the following paragraphs on probabilistic solutions relying on confusion matrices in order to combine classifiers. Thes solutions allow to achieve a form of optimality in regard to objective functions such as expected loss. Let us now formalize classifier combination in terms of probabilistic calculus.

Our goal is to estimate the following conditional aggregated distribution

$$p\left(y|\hat{c}_1(\mathbf{x}),\hat{c}_2(\mathbf{x}),..,\hat{c}_K(\mathbf{x})\right). \tag{1.36}$$

Knowing this distribution, example $\mathbf{x}$ is classified as

$$\hat{c}\left(\mathbf{x}\right) = \underset{y\in\Omega}{\arg\max}\ \ p\left(y|\hat{c}_1(\mathbf{x}),\hat{c}_2(\mathbf{x}),..,\hat{c}_K(\mathbf{x})\right). \tag{1.37}$$

For the sake of equation concision, we will denote by $\hat{c}_k$ either the prediction function learned by the classifier k or the prediction of this classifier for some arbitrary input $\mathbf{x}$.

Applying Bayes rule, the aggregated distribution writes

$$\begin{aligned} p(y|\hat{c}_1,\hat{c}_2,..,\hat{c}_K) &= \frac{p(\hat{c}_1,\hat{c}_2,..,\hat{c}_K|y)}{p(\hat{c}_1,\hat{c}_2,..,\hat{c}_K)}p(y) &(1.38)\\ &\propto p(\hat{c}_1,\hat{c}_2,..,\hat{c}_K|y)p(y). &(1.39)\end{aligned}$$

When K is large, the estimation of the joint conditional distributions $p(\hat{c}_1,\hat{c}_2,..,\hat{c}_K|y)$ is intractable as it involves $(m+1)^K$ parameters. Simplifying hypotheses are needed. Assuming class conditional independence of classifier predictions, we obtain

$$\begin{aligned} p(y|\hat{c}_1,\hat{c}_2,..,\hat{c}_K) &\propto p(\hat{c}_1|y)\times p(\hat{c}_2|y)\times ..\times p(\hat{c}_k|y)\times p(y)\\ &\propto p(y)\prod_{k=1}^{K} p(\hat{c}_k|y). \end{aligned} \tag{1.40}$$

The class distribution $p\left(y\right)$ is a multinomial distribution $Y\sim \text{Mult}\left(\boldsymbol{\pi}\right)$. The vector $\boldsymbol{\pi}$ contains the class probabilities and the maximum likelihood estimate (MLE) for each entry is

$$\pi_i = \frac{n_{\text{val},i}}{n_{\text{val}}}. \tag{1.41}$$

The training set can also be exploited to obtain more accurate estimations. Of course, more elaborate estimates as the MLE can be derived. A prior distribution on $\boldsymbol{\pi}$ can be introduced to obtain a maximum *a posteriori* estimate. Since most datasets usually have the same number of class members, $p\left(y\right)$ is most often a uniform distribution.

Let $\mathscr{P}_\Omega$ denote the simplex of probability distributions over Ω. A **combination rule** in this context is a mapping from $\left(\mathscr{P}_\Omega\right)^K \rightarrow \mathscr{P}_\Omega$. For instance, from an information fusion

perspective, equation (1.40) is a multiplicative conjunctive combination rule applied to classifier performance probabilities which are derived from confusion matrices.

This is the probabilistic baseline approach to fuse classifiers based on performance probabilities. Examples of applications of this approach are given in [43, 94]. In the next chapters, this approach will be referred to as «the Bayes rule approach» and it will be one of the combination methods in the benchmark used in the numerical experiments.

There are, of course, many alternatives to the mainstream approach. In the following paragraphs, we introduce other schemes that may be applied for combining classifiers. In particular, we present probability distribution combination methods and the Bayesian classifier fusion technique. For both categories, we review the state-of-the-art and highlight some interesting contributions.

Probabilistic opinion pooling

The problem of retrieving a consensus probability distribution from a panel of distributions is not new and the first investigations in this field date back to Laplace [71]. An overview of the state-of-the-art on probability distribution combination rules is given in a review by French [35] and in the book of Lee [73]. The construction of combination rules is in general justified by means of desirable properties for the rules. Such properties are for instance the ability of the combination to commute with marginalization or with plugging a prior using Bayes theorem. Let f denote a some probability combination rule. Two popular classes of probability distribution combination rules are:

- logarithmic opinion pools: $f(p_1,..,p_K) \propto \prod_{k=1}^{K} p_k^{a_k}$, $a_k \geq 0$,
- linear opinion pools: $f(p_1,..,p_K) = \sum_{k=1}^{K} a_k p_k$, $a_k \geq 0$ and $\sum_{k=1}^{K} a_k = 1$

Where each a_k is a scalar. In the logarithmic opinion pool, distribution multiplications should be understood as element-wise multiplications. Some extensions of these rules have been proposed in the literature. For instance, Ranjan *et al.* [82] reformulated the linear opinion pool by inserting the cumulative density function of a beta distribution $B_{\alpha,\beta}$ as follows: $f(p_1,..,p_K) = B_{\alpha,\beta} \sum_{k=1}^{K} a_k p_k$ and standard linear pooling is retrieved for $\alpha = \beta = 1$. The authors claim that their Beta linear pooling achieves better performances than standard linear pooling in the estimation of the probability of precipitation forecasts. Other existing probabilistic combination approaches are given in [4].

The set of probability distributions $\{p_k\}_{k=1}^K$ to be pooled is a general set of probabilities and it is not restricted to certain type of distributions. In our classifier combination problem, these distributions are the predictive probabilities $\{p(y|\mathbf{x},\boldsymbol{\theta_k})\}_{k=1}^K$ or the conditional probabilities drawn from confusion matrices like $(\{p(y|\hat{c}_k)\}_{k=1}^K$ or $\{p(\hat{c}_k|y)\}_{k=1}^K$. These latter represent different types of contextual information.

Obviously, the multiplicative rule ($\prod_{k=1}^K p_k$) is a member of the logarithmic opinion pools. In both these classes, the weights a_k capture a form of reliability in the estimates of the distributions p_k. Unless one has prior knowledge concerning the weights, these are not easy to determine. One can estimate them from data [64, 45] but this means that the validation set will be split again. In addition, the split used to obtain weights a_k must contain a sufficient number of data points to jointly estimate K parameters. This is a form of stacking.

Bayesian classifier combination

The Bayesian framework is an attractive way to combine classifiers since it has proved to lead to more accurate and robust estimates in many applications. Kim and Ghahramani [57] adapt a probabilistic model introduced by Dawid and Skene [17] for classifier output fusion under independence assumption. The classification output $\hat{c}_k$ of the k^{th} classifier is a random variable and the conditional distribution of $\hat{c}_k$ given $Y=y$ is multinomial: $\hat{c}_k|y \sim \text{Mult}\left(\boldsymbol{\theta}_y^{(k)}\right)$. In other words, the performance probabilities are the parameters $\theta_{y,i}^{(k)}$ and their estimations derived from the confusion matrices are their MLE estimates. Let $\mathscr{D}_{\text{agg}}$ denote the dataset whose elements are tuples $\left(\hat{c}_1\left(\mathbf{x}^{(i)}\right),..,\hat{c}_K\left(\mathbf{x}^{(i)}\right),y^{(i)}\right)$ for $\left(\mathbf{x}^{(i)},y^{(i)}\right) \in \mathscr{D}_{\text{val}}$. Under classifier independence assumption, the likelihood writes as

$$p\left(\mathscr{D}_{\text{agg}}|\boldsymbol{\theta}_1^{(1)},..,\boldsymbol{\theta}_m^{(K)},\boldsymbol{\pi}\right) = \prod_{i=1}^{n_{\text{val}}} \pi_{y^{(i)}} \prod_{k=1}^{K} \boldsymbol{\theta}_{y^{(i)},\hat{c}_k(\mathbf{x}^{(i)})}^{(k)}. \tag{1.42}$$

The authors propose a Bayesian treatment consisting of using hierarchical conjugate priors on the parameters of all conditional distributions $p(c_k|y)$ as well as on the class distribution $p(y)$. The conjugate priors for $\boldsymbol{\theta}_y^{(k)}$ and for $\boldsymbol{\pi}$ are Dirichlet: $\boldsymbol{\theta}_y^{(k)} \sim \text{Dir}\left(\boldsymbol{\alpha}_y^{(k)}\right)$ and $\boldsymbol{\pi} \sim \text{Dir}(\boldsymbol{\beta})$. A second level of priors is proposed for the parameters $\boldsymbol{\alpha}_y^{(k)}$. The conjugate prior distribution of each $\boldsymbol{\alpha}_y^{(k)}$ is exponential. Gibbs and rejection sampling are then necessary to infer these parameters.

Finally, Kim and Ghahramani extend this model in order to take into account dependencies between classifiers. They propose to use a Markov random field as model of classifier output

interactions. The main limitation of this method is the high computational cost caused by MCMC and rejection sampling. Other related Bayesian approaches are reviewed in [15].

Another approach was introduced by Lacoste *et al.* in [70]. They have a pragmatic vision of the problem where one looks for the best classifier in terms of generalization performances instead of the one that best explains Y given $X = \mathbf{x}$. Let Z denote the random variable representing the index of the best classifier $c_\star$. The best classifier is the risk minimizer:

$$c_\star = \underset{1\leq z\leq K}{\arg\min} \quad R(c_z), \tag{1.43}$$

$$R(c) = \mathbb{E}_{X,Y}\left[L(c(\mathbf{x}), y)\right], \tag{1.44}$$

where L is a loss function. Instead of deriving $p(y|\hat{c}_1,..,\hat{c}_K)$, they propose to focus on inferring Z and then obtain the following predicting distribution:

$$p(c_\star(\mathbf{x}) = y|\mathbf{x}) = \sum_{1\leq z\leq K} p(Z = z)\, p(c_\star(\mathbf{x}) = y|\mathbf{x}, z), \tag{1.45}$$

$$= \sum_{1\leq z\leq K} p(Z = z)\, \mathbb{I}_y(c_z). \tag{1.46}$$

This distribution is obtained by marginalizing out Z and the estimated class for example $\mathbf{x}$ is the one maximizing this distribution. In order to apply this decision function, we must estimate $p(Z = z)$ from the validation set. To that end, let us also consider the 0-1 loss $L_{z,i}$ incurred by choosing classifier c_z and applying it to example $\mathbf{x}^{(i)}$ from $\mathscr{D}_{\text{val}}$. This loss can be regarded as a Bernoulli random variable with parameter $R(c_z)$. By defining priors on the losses and on the risks, standard Bayesian inference allows to estimate the distribution of the risks given all losses observed in $\mathscr{D}_{\text{val}}$. Finally, $p(Z = z)$ is obtained by Monte Carlo approximation. If ones draws n_s samples of the risks given the losses, the probability that $Z = z$ is approximately the number of times that c_z achieves minimum risk over n_s.

The approach of Lacoste *et al.* is intuitive and also guided by classifier performances in terms of risks instead of confusion matrices. It is less computationally demanding than the models proposed by Kim and Ghahramani. In these seminal works, we aim at introducing a classifier combination method with lower computational complexity and in particular that does not rely on any Monte Carlo step.

1.8.2 Evidential classifier combination

The theory of belief functions, a.k.a evidence theory of Dempster-Shafer [21, 87] (D-S) theory is a welcoming playground to build ensemble methods as it generalizes probability

and set theory while providing a number of aggregation operators. Besides, it also generalizes possibility theory which will be our combination framework in the next chapter. In the DS framework, uncertainty about the outcome of events is most often rendered using mass functions which can be understood as a way to encode (lower and upper) bounds on probabilities. The information carried by confusion matrices can also be used to build a mass function for each classifier. After combining these latter, decision making may also be made via expected utilities relying on the Choquet integral of capacities[5] that are in bijective correspondence with the mass function although this is not the only option.

Xu's *et al.* approach [101] is one of the very first methods introducing classifier combination in the belief function framework. Mass functions were the same across all classes for each classifier. For each base classifier, the accuracy and the error rate are used as the respective masses assigned to $\hat{c}_i(\mathbf{x})$ and to $\Omega \setminus \{\hat{c}_i(\mathbf{x})\}$. The remaining mass is assigned to Ω (ignorance) as the base classifiers have optionally a reject class. Mass functions are then combined using Dempster's rule of combination[6]. A set of decision functions involving a thresholding procedure are also presented in the paper for decision making.

Another interesting approach is the one introduced by Rogova [84]. The author combines several neural networks within the D-S theory using information about the classification power of a classifier, i.e. the distance (proximity measure) between the output vector (predicted class probabilities) and reference vectors. Each vector is a mean computed across outputs of a classifier fed with validation samples belonging to a given class. For each class c_j, Rogova builds two mass functions from each proximity measure (one supporting $Y = c_j$ the other supporting $Y \neq c_j$). After combining these $2m$ mass functions, the resulting belief function is a probability distribution. The choice of the distance function and the computation of the reference vector is not obvious. A similar idea of combining classifiers based on a proximity measure between the output and a reference vector is also proposed in [3].

More recently, Burger *et al.* [12] introduced a multi-label SVM obtained by fusing multiple binary SVM classifiers. They utilized belief functions to partially mitigate a loss of information that they argue to occur when combining binary SVMs using voting systems. They build belief functions from fuzzy sets. For each binary SVM and each class, full class membership is assumed outside of the SVM margin while membership progressively decreases as drifting apart from the margin border. The belief functions are combined using Dempster's rule so the relative confidence of each classifier is taken into account in the decision making.

[5] A capacity is a non-additive probability measure.

[6] A popular aggregation operator for belief functions introduced and justified by Dempster himself [21].

Reformat and Yager [83] also transform classifier outputs into belief functions. They use criteria such as accuracy from training data and from validation data. The mass functions thus obtained are combined using Dempster's rule and an ordered weighted averaging operator.

Another approach by Quost *et al.* [81] is also worth mentioning. The authors proposed to combine classifiers based on a rule that is borrowed from fuzzy set theory, i.e. the t-norm rule, and they resolve the combination problem within the belief function theory framework. They build mass functions from classifier outputs and then combine these functions using a parametric t-norm based rule. They propose to determine the parameter λ of the t-norm using model selection via a validation set. In the next chapter, we introduce a new combination method that shares some aspects of Quost *et al.* approach:

- we combine possibility distributions and every such distribution is by definition a special type of belief function,
- we also use a parametric t-norm to combine these distributions,
- we also tune the parameter of the t-norm using a validations set.

However, there also several discrepancies:

- they cluster the classifiers according to their dependencies then they learn one λ for each cluster combination and another λ for merging the mass functions obtained from each cluster while (for now) we only examined batch combination schemes,
- the t-norm is an ingredient to derive a combination rule in the sense of belief functions but it is not a combination rule of its own,
- we build specifically possibility distributions from confusion matrices while Quost *et al.* convert the output of probabilistic classifiers in belief functions,
- they combine separable belief functions which, in general, are not formally equivalent to possibility distributions.

Quite recently in [22], class wise precision and recall (obtained from confusion matrices) are taken as basic ingredients to build mass functions in the framework of belief function theory. Dempster's rule is used to combine all mass functions (for each class and each base classifiers) and the final mass function is transformed into a probability distribution for decision making.

A limitation of the theory of belief functions is computational complexity as the domain of mass functions is the power set 2^{Ω} while both probability and possibility values can be computed from distributions whose domain is Ω.

1.8.3 Fuzzy classifier combination

Fuzzy set theory [103, 104] was designed by Lotfi Zadeh in the mid 60's. In his founding paper [103], Zadeh defined a fuzzy set as «a class of objects with a continuum of grades of membership». Thus, he replaced the dominant binary philosophy, i.e. an element belongs or not to a set, with a gradual philosophy. In other words, each element in a fuzzy set has a membership value determining to what extent the element belongs to the set in contrary to «crisp sets» that include elements whose membership functions are equal to 0 or 1. Fuzzy set theory was conceived to present «soft» classification of elements which is more adapted to the way people create categories in natural language. Thanks to Fuzzy set theory, possibility theory [105] was later introduced to handle incomplete information in natural language.

Classifier combination has also been examined in the fuzzy set theory community. Cho and Kim [14] introduce a method for combining multiple neural networks using fuzzy integrals. For each class, the output scores of the neural nets are used to build a function which is integrated using the fuzzy integral w.r.t a fuzzy measure drawn from each individual classifier performance on the training set. After m such integrations, the predicted class is the one with the highest integrated measure. The fact that the training set leads to optimistic accuracy estimation is not discussed.

In [56], Keller *et al.* present a similar approach. However, they consider more general fuzzy integrals in which usual min and max operators are replaced with a t-norm and a t-*conorm*. The two previous approaches rely on base classifiers that can deliver classification scores, e.g. probabilistic classifiers.

In line with above approaches (but tailored for a given application), Gader *et al.* [38] proposed a method for combining three classifiers for a handwritten word recognition task. The authors combined the outputs of a hidden Markov model with those of a fuzzy hidden Markov model and a segmentation-based classifier. They used the ranking of the top 5 possible lexicons produced by each classifier to heuristically compute the density[7] functions that are necessary to compute fuzzy integrals which are the decision functions. Choquet integral reported higher classification accuracy as compared to Sugeno integral based fusion or other rank-based combination schemes such as Borda and weighted Borda counts.

Another classical work relying on a fuzzy set theoretic step is the decision template approach [68]. It also relies on base classifiers yielding classification scores. If s_i denotes the score functions for the i^{th} classifier, the vector $[s_1(\mathbf{x}), \ldots, s_K(\mathbf{x})]^t$ is called the decision profile of the example $\mathbf{x}$. At training time, the class wise averages of profiles are computed

[7] in the sense of fuzzy measures

and are called decision templates. At test time, the predicted class is the one for which the similarity between the profile of the test sample and one of the m templates is maximal. If scores functions are seen as membership functions, then similarities can be computed using aggregation operators for fuzzy sets.

The usefulness of fuzzy fusion for AdaBoost ensemble classifiers was studied in [66]. The author recursively constructs an AdaBoost ensemble classifier by computing fuzzy densities which, when fixed, are used to find fuzzy integrals to take a final decision. Multilayer neural nets with one hidden layer were employed as elementary classifiers during an experimental comparison of fuzzy and non-fuzzy combination algorithms. The results demonstrate the benefits of fuzzy integrals in the combination allowing to yield better performances than those of individual classifiers. Again, boosting is a particular form of aggregation which is different from the problem we address in this thesis in which we cannot act on the base classifiers which are taken as granted.

1.9 Conclusion

This chapter constitutes the state-of-the-art of this thesis. First, we stated the classification problem and briefly reviewed some learning algorithms addressing this problem. We then expressed some general motivations for combining classifiers. The major motivations concerns statistical, representational and computational limitations as explained in section 1.3. These limitations may be prevented by a classifier aggregation. In addition, there are other motivations about the usefulness of combination based solutions like breaking complex classification problems in simpler ones and also like alleviating the difficulty of choosing a classification algorithm without prior knowledge. In sections 1.5, 1.6, 1.7 and 1.8, we presented some well known combination methods that are summarized in table 1.4. The majority of them rely on base classifiers that are trained independently which is also the case for our contributions in the following chapters where we develop classifier fusion methods within two different uncertainty frameworks.

We decided to follow a similar track as the baseline probabilistic combination approach presented in subsection 1.8.1 that consists in estimating the joint conditional distribution (eq 1.36). However, without independence assumption, the problem involves many parameters which is computationally costly and may cause overfitting. But if we assume independence, statistical dependence in classifier decisions is unfortunately neglected.

In this thesis, we intend to design novel combination methods that take advantage of confusion matrices and in particular of the conditional distributions issued from these

matrices. The proposed approaches are also conceived to handle statistical dependence between classifiers which has several roots such as correlated training samples or the usage of classifiers of the same type i.e. homogeneous classifier combination. Thus, our original contributions are introduced in chapters 2 and 3 where two novel techniques are developed.

In the chapter 2, we propose to carry over the combination problem to the possibilistic framework. To that end, each conditional probability distribution is transformed into a possibility distribution. In possibility theory, many combination rules are available in the literature and they do not involve computational difficulties. In particular, we investigate parametrized t-norm based rules. The t-norm parameter is denoted λ and is a hyperparameter. It is a scalar which we set using cross validated grid search to regulate the level of dependency between classifiers.

In the chapter 3, we overcome the computation of the joint conditional distribution using copula functions which are statistical models that handle dependencies between variables. We take advantage of Sklar's theorem which compute the joint conditional distribution out of marginal conditional probabilities (from confusion matrices). We investigate Gaussian copulas because they allow tractable combinations even if K is large. Similarly as for the possibilistic approach, Gaussian copulas have a single hyperparameter that regulates the level of dependencies between the variables $\hat{c}_k|y$ and this hyperparameter is also tuned by grid search.

Combination approaches	Parametric	Types of classifiers	Uncertainty framework	Base classifier training	Contextual data	Base classifier prerequisites	learning of the combination rule
Voting	no	homo/hete	deterministic	independent	none	none	no
Borda counts	both	homo/hete	deterministic	independent	none	none	no
BKS [51]	yes	homo/hete	deterministic	independent	confusion matrices	none	no
Bagging [7]	no	homo	deterministic	independent	none	none	no
Boosting [85]	yes	homo	deterministic	sequential	none	none	yes
Random subspaces [47]	no	homo	deterministic	independent	none	none	no
ECOC [26]	no	homo	deterministic	independent	none	none	no
Stacking [95]	no	homo/hete	deterministic	independent	no	non	yes
Mixture models [13]	yes	homo/hete	probabilistic	independent	no	probabilistic	yes
Mixture of experts [49]	yes	homo/hete	probabilistic	independent	no	probabilistic	yes
Bayes rule combination	no	homo/hete	probabilistic	independent	confusion matrices	none	no
Probabilistic opinion pooling [35]	yes	homo/hete	probabilistic	independent	none	none	no
Xu et al. [101]	no	homo/hete	Belief function theory	independent	accuracy and error rates	none	no
Cho and Kim [14]	no	probabilistic	Fuzzy set theory	independent	accuracy rate	probabilistic	no

Table 1.4 Summarizing table for some combination approaches reviewed in this chapter.

Chapter 2

Possibilistic t-norm based combination

2.1 Introduction

The aim of this PhD is to develop new classifier fusion approaches in order to obtain better generalization performances especially in situations where the dataset is distributed (as in the decentralized learning setting) and we cannot make assumptions on the mechanisms governing the partitioning of the data. To achieve this task, we propose to follow two principles:

- make use of contextual information (either provided or estimated from a validation set) in the combination scheme. We will typically use such information to derive confidence level that can be given to an individual classifier within a panel.
- adapt the combination rule to the dependence between classifiers.

The type of contextual information considered in this manuscript is the one contained in confusion matrices or related classifier performance pieces of information. Concerning dependency, most of existing approaches rely on classifier independence assumptions as a prerequisite. However, in practice, this assumption is very unlikely to be verified as classifiers tend to perform well in the same region of the input space. When two classifiers are highly dependent, they may excessively influence the combination result toward a given solution. One solution to circumvent unjustified overconfidence is to resort to idempotent combination operators. In the framework of possibility theory, the entrywise minimum of possibility distributions is an idempotent operator. In this chapter, we propose a combination approach that consists in carrying over the classifier fusion problem to the possibilistic framework. In this framework, we examine t-*norm* based combination for possibility distributions as a

flexible class of models that are robust to classifier dependence. We demonstrate through several experiments that our possibilistic combination approach achieves competing generalization performances as compared to reference methods. We also include a classifier fault tolerance study at the end of this chapter in which the added value of the proposed approach is highlighted. Also, it should be stressed that although we discuss probabilistic or possibilistic aggregation of classifiers we do not require that classifiers are also themselves probabilistic or possibilistic therefore the combination mechanism considered applies to any type of base classifier. Also, for the sake of interpretability of the numerical results, we carry out our experiments on homogeneous ensembles but the approach is not limited to this setting and can also combine heterogeneous ensembles.

In the following section, we further justify the benefits of possibilistic approaches in the problem of classifier combination relying on classifier individual performances in parallel architectures. Section 2.3 presents the possibility theory framework and the proposed t-*norm* based possibilistic classifier combination. Section 2.4 contains several numerical experiments. For seven different standard public datasets, we compare the aggregation performances for three different classifier ensembles.

2.2 Combination problem statement

Our goal is to reconcile the predictions delivered by our ensemble of classifiers. If we picture the set of those predictions as a vector, we can build a second stage classification step, or meta-classifier that falls in the scope of the problem stated in the previous chapter. Indeed, under 0-1 loss, the optimal decision rule consists in maximizing the following probabilities w.r.t. y

$$p\left(y|\hat{c}_1(\mathbf{x}),\hat{c}_2(\mathbf{x}),..,\hat{c}_K(\mathbf{x})\right). \tag{2.1}$$

If $\hat{c}$ denotes the aggregated prediction function, example $\mathbf{x}$ is classified as

$$\hat{c}\left(\mathbf{x}\right) = \underset{y\in\Omega}{\arg\max}\ \ p\left(y|\hat{c}_1(\mathbf{x}),\hat{c}_2(\mathbf{x}),..,\hat{c}_K(\mathbf{x})\right). \tag{2.2}$$

We could think of estimating $p\left(y|\hat{c}_1,\hat{c}_2,..,\hat{c}_K\right)$ based on the validation set just as the distributions $\{p(\hat{c}_k|y)\}_{k=1}^{K}$ (or confusion matrices) are estimated. However, this frequentist estimation problem is much more challenging. Indeed, each such distribution has m parameters and there are m^K such distributions. So we see that the dimensionality of the problem does not scale well in both m and K.

Applying Bayes formula, we have

$$p\left(y|\hat{c}_1(\mathbf{x}),\hat{c}_2(\mathbf{x}),..,\hat{c}_K(\mathbf{x})\right) \propto p\left(\hat{c}_1(\mathbf{x}),\hat{c}_2(\mathbf{x}),..,\hat{c}_K(\mathbf{x})|y\right) \times p\left(y\right). \tag{2.3}$$

Class probabilities can be derived from the confusion matrices but again, the estimation of conditional joint distributions $p\left(\hat{c}_1(\mathbf{x}),\hat{c}_2(\mathbf{x}),..,\hat{c}_K(\mathbf{x})|y\right)$ has the same complexity as the estimation of the posterior.

Linear complexity can be achieved by making conditional independence assumptions that allow each conditional joint distribution to factorize as the product of its marginals, that is

$$p\left(y|\hat{c}_1(\mathbf{x}),\hat{c}_2(\mathbf{x}),..,\hat{c}_K(\mathbf{x})\right) \propto p\left(y\right) \times \prod_{k=1}^{K} p\left(\hat{c}_i(\mathbf{x})|y\right). \tag{2.4}$$

In this approach, which will be referred to as the Bayes rule approach, we see that the probabilities derived from confusion matrices allow to derive the aggregated classification rule. Unfortunately, the independence assumptions are obviously not justified because the classifiers output are highly correlated. Indeed, examples that are difficult to classify correctly for classifier $\hat{c}_i$ is usually also difficult to classify correctly for classifier $\hat{c}_j$. This means that the aggregate classifier will this time underfit. The dependence between classifiers has its roots in several causes like for instance learning on shared examples, use of classifiers of the same type, correlation between training examples. This accounts for the fact that misclassifications for each $\hat{c}_k$ occur most of the time with the same inputs. This observation also motivated the derivation of boosting since this technique tries to circumvent this issue by building (incrementally) a new classifier focusing on correctly classifying examples that were misclassified by the others.

The above analysis justifies the need for models with higher capacity than the Bayes rule approach but less than the frequentist approach. In this scope, an idea is to use a probability distribution pooling model [35], i.e. to choose a parametric combination rule $f_{\boldsymbol{\theta}}$ such that

$$p\left(y|\hat{c}_1(\mathbf{x}),\hat{c}_2(\mathbf{x}),..,\hat{c}_K(\mathbf{x})\right) = f_{\boldsymbol{\theta}}\left(p(y|\hat{c}_1),..,p(y|\hat{c}_K)\right), \tag{2.5}$$

where $\boldsymbol{\theta}$ is a vector of parameters which can be estimated using the validation set.

As outlined in 1.8.1, there are two main probability distribution combination model: the linear and the logarithmic opinion pool. Each of these models is justified by a number

of axiomatic properties. In both cases, the dimensionality of the parameter space is K. There are algorithmic solutions [64, 45] in the literature allowing to learn parameters $\boldsymbol{\theta}$ in a machine learning context. However, these algorithms are either suboptimal or involve costly optimization steps.

In this chapter, we intend to derive an approach that achieves better computational tractability. We propose to carry over the combination problem to the possibilistic framework. To that end, each conditional probability distribution $p(y|\hat{c}_k(\mathbf{x}) = c_i)$ is transformed into a possibility distribution $\pi^{(k,c_i)}(y)$ where $c_i \in \Omega$. In possibility theory, many combination rules are available in the literature and they do not involve computational difficulties. In particular, we investigate parametrized t-*norm* based rules. The t-*norm* parameter is denoted λ and is a hyperparameter. It is a scalar which we set using cross validated grid search. Having set λ, the ensemble consensus possibility distribution writes

$$\pi^{(\text{ens},\mathbf{x})}(y) = \mathrm{T}_\lambda \left(\pi^{(1,\hat{c}_1(\mathbf{x}))}(y), \ldots, \pi^{(K,\hat{c}_K(\mathbf{x}))}(y) \right), \tag{2.6}$$

where T_λ denotes a t-norm. The range of values for λ allows to visit different models with variable treatments of input distribution dependency.

But since $\pi^{(\text{ens},\mathbf{x})}$ is not a probability distribution, we need to use another decision theoretic setting than the minimization of expected loss against the posterior distribution. Gilboa [42] showed that expected loss can also be computed in terms of Choquet integral against non additive monotone measures (i.e. capacities). Following this philosophy, we will use the following decision rule

$$\hat{c}(\mathbf{x}) = \underset{y \in \Omega}{\arg\max} \ \pi^{(\text{ens},\mathbf{x})}(y). \tag{2.7}$$

It is noteworthy that the range of values visited for λ allows the proposed approach to fulfill our requirement of trade-off learning capacity as compared to the Bayes rule and frequentist approaches. In particular, for the t-*norm* family considered later in section 2.3, one value of λ achieves t-*norm* idempotence which offers a correct treatment of the maximal dependency situation in which each $\hat{c}_i$ are K copies of the same decision function. Indeed, in this case we have $\pi^{(\text{ens},\mathbf{x})} = \pi^{(1,\hat{c}_1(\mathbf{x}))} = \ldots = \pi^{(K,\hat{c}_K(\mathbf{x}))}$ and we make our decisions as if we had only one base classifier. Other values will capture lower levels of dependencies between the base classifiers.

The proposed method is presented in details (nature of the transformation and t-*norms*) in the next section.

2.3 T-norm based possibilistic combination of classifiers

In this section, we give a detailed presentation of the proposed possibilistic approach to dependent classifier combination. As previously mentioned, suitable t-*norm* rules for possibility measures combination will be used to aggregate classifier decisions based on their estimated performances. We first recall basic notions of possibility theory as an uncertainty representation framework and its connections with probability theory, then we present the t-*norm* based combination process.

2.3.1 Basic background on possibility theory

Possibility theory was introduced by Zadeh in [102] and further developed by Dubois and Prade [29] with the motivation to offer a well-defined and formal mathematical representation for linguistic statements that permits handling imprecise or vague information. For instance, the word *cheap* can be given a large set of values according to everyone's subjective definition and context of cheapness. Possibility values can be interpreted as *degrees of feasibility* of event occurrence. An important difference with probability theory is that high possibility values are non-informative while high probability values are. Indeed, a very high possibility for event A means that, should A occur or not, we would not be surprised. If A has a very high probability mass, then we would surprised that A does not occur. Conversely, low possibility and probability values are both informative as they both indicated that an occurrence of A is unlikely.

Possibility theory is also related to the theory of belief functions as it can be proved that if a mass function is consonant (i.e. it has nested focal elements), then it is in bijective correspondence with a possibility distribution. More generally, possibilities are a special class of imprecise probabilities which is a framework in which probability values can only be bracketed by two bounds. Indeed, the uncertainty of an event in the possibilistic framework is better upraised by a pair of values (possibility and necessity) which can be seen as probability bounds.

Possibility theory has its roots in fuzzy set theory. Indeed, suppose $\mathscr{T}$ is the set of events $B \subseteq \Omega$ that are true and $\mathscr{U}$ is the set of undecided events (those that are neither true nor false). Suppose also that $\mathscr{T}$ and $\mathscr{U}$ are fuzzy, then the necessity is the membership function of $\mathscr{T}$ and the possibility is the membership function of $\mathscr{T} \cup \mathscr{U}$.

In the classifier combination approach introduced as part of this Ph.D., we are in line with the imprecise probability theoretic interpretation of possibilities and several comments in the remainder of this chapter are a consequence of this choice.

Possibility distributions

In possibility theory, possibility distributions are the simplest class of objects that entirely capture all information on our uncertainty. A possibility distribution π maps each element in the universe of discourse to the unit interval [0,1] whose extreme values correspond to impossible and totally possible states. The set of admissible probability distributions with the upper and lower bound constraints induced by the possibility π is denoted by $\mathscr{P}_\pi \subseteq \mathscr{P}_\Omega$.

If any state $c_i \in \Omega$ has a possibility degree equal to 1, then this state (i.e. this class), is totally possible and by convention the possibility distribution π is said to be normalized. The imprecise probability interpretation is only valid if possibility distributions are normal otherwise we would have $p(\Omega) < 1$ for some probability distribution p in $\mathscr{P}_\pi$. We can distinguish two particular situations of zero and full certainty when dealing with a possibility distribution:

1. Full certainty: $\exists c_i \in \Omega$, $\pi(c_i) = 1$ and $\pi(c_j) = 0$, $\forall\, c_j \neq c_i$.
2. Zero certainty (ignorance): $\pi(c_i) = 1$, $\forall c_i \in \Omega$.

For the sake of equation concision, we will use π_i to denote $\pi(c_i)$ when necessary.

Possibility and necessity measures

A typical aspect of possibility theory and other theories compatible with imprecise probabilities is the presence of two measures to describe uncertainty: necessity and possibility. A *necessity measure* accounts for justified degrees of belief on each event in light of the available information. The corresponding *possibility measure* evaluates to what extent one can still say that an event is possible in the absence of any contradictory information. The necessity and possibility measures are respectively the lower and upper probabilities in the imprecise probability interpretation. Given a subset A of Ω, a possibility measure is given by:

$$\Pi(A) = \max_{c_i \in A} \pi(c_i) \tag{2.8}$$

which means that the possibility of a subset A is equal to the maximum possibility degree in this subset. A possibility measure is thus maxitive: $\Pi(A \cup B) = \max(\Pi(A), \Pi(B))$ as opposed to probability measures which are summative. Observe that this property accounts for the fact that the possibility distribution is enough information to compute the possibility measure of any subset.

The necessity measure is given by:

$$N(A) = 1 - \Pi(\bar{A}) = \inf_{c_i \notin A} (1 - \pi(c_i)) \tag{2.9}$$

Where $\bar{A}$ is the complement of A in Ω. The necessity measure is such that: $N(A \cap B) = \min(N(A), N(B))$ and, in the normalized case, we have that $\Pi(\Omega) = N(\Omega) = 1$ and $\Pi(\emptyset) = N(\emptyset) = 0$.

Probability-Possibility transformation

As justified in subsection 2.2, we need to convert conditional probability distributions into possibility distributions in order to achieve our classifier fusion goal. Remembering that the distributions in question are objective probabilities, the recommended choice is a transformation due to Dubois and Prade [28] which preserves the statistical information contained in the probabilities.

Considering a discrete probability distribution p on Ω, we can always permute the indexes of the elements of Ω such that the set of probability values is sorted in descending order: $p_i \geq p_{i+1}$ for any $i < m$ and where p_i denotes $p(c_i)$. The transformation reads

$$\pi_i = \begin{cases} 1 & \text{if } i = 1 \\ \pi_{i-1} & \text{if } i > 1 \text{ and } p_i = p_{i-1} \\ \sum_{j=i}^{m} p_i & \text{otherwise} \end{cases}. \tag{2.10}$$

This transformation is reversible [30] (in the sense that p can recovered from π). It produces a normalized possibility distribution. If p is uniform, then p is mapped to a constant one π. If p is a Dirac mass then p is mapped to itself. It also has three important properties [27]:

- *consistency*: $\forall A \subseteq \Omega$, $\Pi(A) \geq p(A)$ where Π is the possibility measure spanned by π. So Π is a well defined upper probability.
- *preference preservation*: $\forall (c_i, c_j) \in \Omega^2$, $p_i > p_j \Leftrightarrow \pi_i > \pi_j$, so there is a form of compatibility between the preferences encoded by π and those encoded by p.
- *maximal specificity*: π achieves maximal specificity among those possibility distributions consistent and preserving preferences with p. Considering two possibility

distribution $\pi^{(1)}$ and $\pi^{(2)}$, the possibility distribution $\pi^{(1)}$ is said to be more informative than $\pi^{(2)}$, denoted $\pi^{(1)} \preceq \pi^{(2)}$, if $\forall i \in \{1;\ldots;m\}$, $\pi^{(1)}(c_i) \leq \pi^{(2)}(c_i)$. We also have $\pi^{(1)} \preceq \pi^{(2)} \Leftrightarrow \mathscr{P}_{\pi^{(1)}} \subseteq \mathscr{P}_{\pi^{(2)}}$ which offers an intuitive justification regarding the ability of relation $\preceq$ to capture informational content levels.

In the sequel, we will denote by $\pi^{(k,c_i)}(y)$ the possibility distribution obtained from $p(y|\hat{c}_k(\mathbf{x}) = c_i)$ via the mechanism described in this subsection.

2.3.2 T-norm based combination

As outlined in 2.2, after obtaining the possibility distributions $\pi^{(k,c_i)}$ (for all $k \in \{1,\ldots,K\}$ and all $i \in \{1,\ldots,m\}$), we need some mean to aggregate these latter in order to obtain a single possibility distribution that will allow us to make a decision as to which class example $\mathbf{x}$ belongs to. This possibility distribution can be regarded as a consensus summarizing the information encoded in each $\pi^{(k,\hat{c}_k(\mathbf{x}))}$. The consensus possibility distribution is denoted by $\pi^{(\text{ens},\mathbf{x})}$. So in this subsection, we essentially examine candidate combination rules in the possibilistic framework allowing to compute $\pi^{(\text{ens},\mathbf{x})}$.

Combining dependent classifier decisions requires extra care. Suppose two classifiers are identical (same training algorithm run on the same dataset) and consequently we have $\hat{c}_k = \hat{c}_{k'}$ for some $k \neq k'$. The solutions supported by this pair of redundant classifiers will have an unjustified impact on the pooled ones. One possibility to circumvent this problem is to resort to idempotent rules which will count functions $\hat{c}_k$ and $\hat{c}_{k'}$ as one.

Conversely, it can also be argued that two learning algorithms trained on different datasets may converge to very close decision functions $\hat{c}_k \approx \hat{c}_{k'}$. In this case, the classifier outputs are also dependent but the fact that they reinforce some candidate solutions appears to be justified because these outputs have their origins in different evidence. This well known information fusion phenomenon calls for an adaptive fusion acting as a trade-off between idempotent rules and rules with reinforcement.

As previously mentioned, we investigate t-*norm* operations T_λ as candidate combination rules:

$$\pi^{(\text{ens},\mathbf{x})}(y) = \mathrm{T}_\lambda\left(\pi^{(1,\hat{c}_1(\mathbf{x}))}(y),\ldots,\pi^{(K,\hat{c}_K(\mathbf{x}))}(y)\right), \forall y \in \Omega. \tag{2.11}$$

This choice is compliant with our goal as it offers a continuum of combination rules involving an idempotent one (minimum rule) and a reinforcing one (multiplicative rule). This coverage is allowed by setting a single hyperparameter λ. Since it is not possible to predict whether indempotence or reinforcement will be beneficial for a given classifier combination problem,

we propose to learn λ from data through a cross-validation procedure. This will allow the selected t-*norm* to capture the appropriate behavior.

In the remainder of this section, we give necessary backgrounds on t-*norms* and present the t-*norm* family we recommend for classifier combination. A triangular norm, or t-*norm*, is a well defined aggregation operator for possibility distributions as it is a mapping defined as follows :

$$\begin{aligned} \mathrm{T}_\lambda :&[0,1]\times[0,1] \longrightarrow [0,1] \\ &(a,b) \longrightarrow \mathrm{T}_\lambda(a,b) \end{aligned} \tag{2.12}$$

This is sufficient to ensure that an element-wise combination of two possibility distributions using a t-*norm* yields a possibility distribution. Note that this distribution may no longer be normalized. We can renormalize the distribution by dividing it by its maximal value. If this maximal value is null, then by convention, the renormalized distribution is constant one (ignorance). In practice, this is not necessary as we maximize the distribution to classify an example. This is only desirable in order to preserve the imprecise probability interpretation.

Besides, A t-*norm* is commutative, associative and has 1 as neutral element. Using these properties, one is able to build a n-ary operator (also denoted T_λ) as

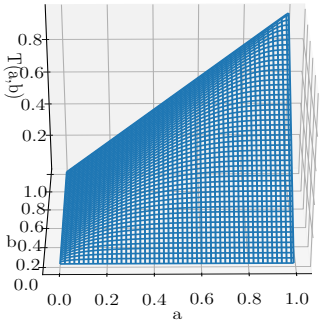
$$\begin{aligned} \mathrm{T}_\lambda :\ & [0,1]^n && \longrightarrow [0,1] \\ & (a_1,..,a_n) && \longrightarrow \mathrm{T}_\lambda(a_1,..,a_n) = \mathrm{T}_\lambda(a_1,..,\mathrm{T}_\lambda(a_{n-1},a_n)) \end{aligned} \tag{2.13}$$

Any t-*norm* also possesses the monotonicity property which writes as follows: for any $a,b,c,d \in [0,1]$ such that $a \leq c$ and $b \leq d$, then $T_\lambda(a,b) \leq T_\lambda(c,d)$.

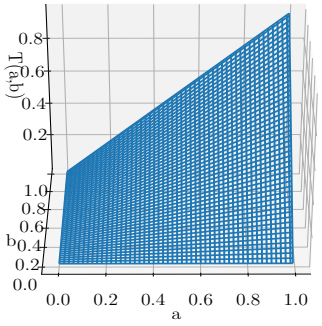
There are many t-*norm* families that are worth being investigated, but in this chapter we limit our choice to **Aczel-Alsina** t-norm family since the differences in performances of different t-norms families are not significant (proof in Appendix A). A non exhaustive list of t-norm families are presented in Appendix A. The mathematical expression for Aczel-Alsina family is given by

$$\mathrm{T}_\lambda(a,b) = e^{-(|\log a|^\lambda + |\log b|^\lambda)^{\frac{1}{\lambda}}}. \tag{2.14}$$

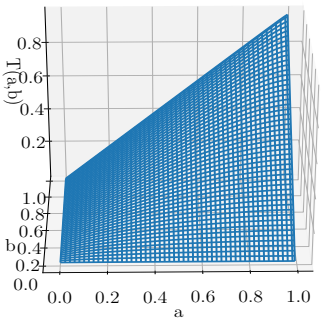
For Aczel-Alsina family, the parameter λ has range $[0;+\infty]$. The limiting rules are respectively the drastic rule (if $a=1, \mathrm{T}_\lambda(a,b)=b$ and if $b=1, \mathrm{T}_\lambda(a,b)=a$ otherwise $\mathrm{T}_\lambda(a,b)=0$) and the minimum rule. We can verify that for $\lambda=1$, we obtain $\mathrm{T}_\lambda(a,b)=ab$ (multiplicative rule). Observe that the Bayes rule and the multiplicative rule approaches coincide only if classes and predicted classes by each classifier have a uniform probability distribution, which has little chance to occur in practice.



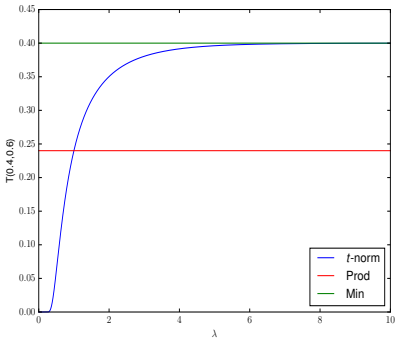
(a) $\lambda \longrightarrow +\infty$: minimum rule



(b) $\lambda \longrightarrow 1$: product rule



(c) $\lambda = 1.5$



(d) $a = 0.4, b = 0.6$

Fig. 2.1 Aczel-Alsina t-*norm* representation with respect to the parameter λ

A 3D-plot of t-*norm* for certain values of λ shows the influence of this parameter on the shape of the mapping T_λ (Figure 2.1). For a given pair (a,b), Figure 2.1d shows that when λ is equal to 1 the t-*norm* intersects the multiplicative rule while the minimum rule is retrieved for large values of λ.

The fact that Aczel-Alsina family leads to tractable and well defined combinations of possibility distributions are not sufficient to qualify it as a legitimate information fusion operation. To that end, we give the following informational content argument: the consensus possibility distribution obtained from Aczel-Alsina t-*norm* combination is idempotent when $\lambda \to +\infty$ and progressively allows confidence reinforcement in predictions as λ decreases. In mathematical terms, if $\pi^{(1)}$ and $\pi^{(2)}$ are two consistent input possibility distributions, i.e.

$\exists c_i \in \Omega$ s.t. $\pi^{(1)}(c_i) = \pi^{(2)}(c_i) = 1$ and $\lambda < \lambda'$, then we have

$$\pi^{(\lambda)} \preceq \pi^{(\lambda')}, \tag{2.15}$$

where $\pi^{(\lambda)}$ and $\pi^{(\lambda')}$ are the consensus distributions obtained respectively from T_λ and $\mathrm{T}_{\lambda'}$. This is an immediate consequence of the fact that when input distributions are consistent there is no need to renormalize and that T_λ is an increasing function w.r.t. λ. In conclusion, the fusion model can select an idempotent rule if the classifiers are extremely dependent or it can deliver a result with increased informational content as we also have

$$\mathscr{P}_{\pi^{(\lambda)}} \subseteq \mathscr{P}_{\pi^{(1)}} \cap \mathscr{P}_{\pi^{(2)}}. \tag{2.16}$$

This follows from the fact that the least informative consensus distribution is achieved for $\lambda = +\infty$ and this distribution is the entrywise minimum of the input distributions.

When input distributions are not consistent (which may happen when classifiers disagree), these results no longer hold because the renormalization might lead to poorly informative consensus distributions. For instance, if input distributions are indicator functions of two different classes, then, after renormalization, $\mathscr{P}_{\pi^{(\lambda)}} = \mathscr{P}_\Omega$. In general, the Aczel-Alsina t-*norm* combination yields a consensus possibility distribution that is a compromise in light of the inconsistent messages encoded by the input distributions. In particular, as a consequence of the monotonicity of t-*norms*, we always have the following preference preservation property regardless of input distribution consistency:

$$\pi^{(1)}(c_i) \geq \pi^{(1)}\left(c_j\right) \text{ and } \pi^{(2)}(c_i) \geq \pi^{(2)}\left(c_j\right) \Rightarrow \pi^{(\lambda)}(c_i) \geq \pi^{(\lambda)}\left(c_j\right). \tag{2.17}$$

2.3.3 Hyperparameter tuning

One of the advantages of the proposed *t*-norm combination rule is the presence of a single parameter to set, making the combination process more computationally tractable than other rules which involve a number of parameters that is linear in the number of classifiers such as linear or logarithmic opinion pool.

To ensure that a relevant value of λ is selected, we perform grid search for parameter λ and retain the value producing maximal accuracy on the validation set. This value is our estimate of λ denoted by $\hat{\lambda}$. This raises the question of the grid definition especially when λ lives in an unbounded interval as in our case. It has been experimentally validated that a grid resolution below 0.1 is not necessary which also accounts for the robustness of the proposed

method. Also, as illustrated in Figure 2.1d , there is no need to investigate very high values for λ as T_λ (as a function of λ) converges rapidly. Once $\hat{\lambda}$ is obtained, we re-train each classifier on the whole dataset (training and validation) and we aggregate their outputs at test time using $T_{\hat{\lambda}}$.

Before moving to the experimental section of this chapter, we summarize our global combination approach in Algorithm 2. A less formal version of the same procedure is also given in Algorithm 3. This algorithm has the following inputs: a training dataset $\mathscr{D}_{\text{train}}$, a validation dataset $\mathscr{D}_{\text{val}}$, a set of supervised learning algorithms $\{\text{train-alg}_k\}_{k=1}^{K}$, a t-*norm* T_λ and a set which is denoted by grid_λ of predefined values for the hyperparameter λ.

Algorithm 2: T-norm based possibilistic combination model (training)

Data: $\mathscr{D}_{\text{train}}$, $\mathscr{D}_{\text{val}}$, $\{\text{train-alg}_k\}_{k=1}^{K}$, T_λ, and grid_λ

for $k \in \{1,\dots,K\}$ **do**
- Run train-alg_k on $\mathscr{D}_{\text{train}}$ to learn $\hat{c}_k$
- Estimate $p(y|\hat{c}_k = c_i)$ using (1.23) and $\mathscr{D}_{\text{val}}$
- Obtain $\pi^{(k,c_i)}(y)$ from $p(y|\hat{c}_k = c_i)$ using (2.10)

for λ *in* $grid_\lambda$ **do**
- **for** $\left(\mathbf{x}^{(i)}, y^i\right) \in \mathscr{D}_{val}$ **do**
  - $\pi^{\left(\text{ens},\mathbf{x}^{(i)}\right)} \leftarrow \mathrm{T}_\lambda\left(\pi^{(1,\hat{c}_1(\mathbf{x}^{(i)}))},\dots,\pi^{(K,\hat{c}_K(\mathbf{x}^{(i)}))}\right)$
  - Obtain $\hat{c}\left(\mathbf{x}^{(i)}\right)$ using (2.7)
  - $\text{Acc}(\lambda) \leftarrow \text{Acc}(\lambda) + \frac{1}{n_{\text{val}}} \times \mathbb{I}_{y^{(i)}}\left(\hat{c}\left(\mathbf{x}^{(i)}\right)\right)$

$\hat{\lambda} \leftarrow \underset{\lambda \in \text{grid}_\lambda}{\arg\max}\ \text{Acc}(\lambda)$

for $k \in \{1,\dots,K\}$ **do**
- Run train-alg_k on $\mathscr{D}_{\text{train}} \cup \mathscr{D}_{\text{val}}$ to learn $\hat{c}_k$

return $\hat{c}_1,\dots,\hat{c}_K,\pi^{(1,1)},\dots,\pi^{(K,m)}$ *and* $\hat{\lambda}$

For more reliable estimates, it is recommended to embed Algorithm 2 in a cross-validation loop to split the initial dataset into $\mathscr{D}_{\text{train}}$ and $\mathscr{D}_{\text{val}}$ as we do in the experiments. After $\hat{\lambda}$ is obtained, re-training the $\hat{c}_k$'s on $\mathscr{D}_{\text{train}} \cup \mathscr{D}_{\text{val}}$ is an optional step. Finally, the selection of $\hat{\lambda}$ in Algorithm 2 is performed by maximizing the estimated accuracy $\text{Acc}(\lambda)$ but other performance criteria could be used instead.

Algorithm 3: T-norm based possibilistic combination model (training)

Data: $\mathscr{D}_{\text{train}}$, $\mathscr{D}_{\text{val}}$, $\{\text{train-alg}_k\}_{k=1}^{K}$, T_λ, and grid_λ

for $k \in \{1,\dots,K\}$ **do**
  Run train-alg$_k$ on $\mathscr{D}_{\text{train}}$ to learn $\hat{c}_k$
  Estimate probability distribution $p(y|\hat{c}_k = c_i)$ using confusion matrices
  Obtain possibility distributions $\pi^{(k,c_i)}(y)$ using Dubois Prade (2.10)
for λ *in* $grid_\lambda$ **do**
  for $\left(\mathbf{x}^{(i)}, y^i\right) \in \mathscr{D}_{val}$ **do**
    Combine possibility distributions $\pi^{(k,c_i)}(y)$ using T_λ to obtain $\pi^{\left(\text{ens},\mathbf{x}^{(i)}\right)}$
    Obtain $\hat{c}\left(\mathbf{x}^{(i)}\right)$ using (2.7)
    Compute accuracy Acc(λ)
$\hat{\lambda} \leftarrow \underset{\lambda \in \text{grid}_\lambda}{\arg\max}\ \text{Acc}(\lambda)$
for $k \in \{1,\dots,K\}$ **do**
  Run train-alg$_k$ on $\mathscr{D}_{\text{train}} \cup \mathscr{D}_{\text{val}}$ to learn $\hat{c}_k$
return $\hat{c}_1,\dots,\hat{c}_K, \pi^{(1,1)},\dots,\pi^{(K,m)}$ *and* $\hat{\lambda}$

2.4 Application on real datasets

This section contains several experimental results assessing the benefits of our possibilistic classifier combination approach. We also comment on the statistical validation of the results.

2.4.1 Datasets

To demonstrate the effectiveness of the proposed possibilistic aggregation method, we tested it on 7 datasets from the UCI repository and the Scikit-learn python library. They exhibit differences in the number of examples, number of classes and in the number of features. The selected datasets are: Digits, Waveform, Cancer, CNAE, MNIST, Wine, Segments. Table.2.1 gives more details on their specificities. Repeated performance discrepancies across heterogeneous datasets are featuring the ability of an ensemble method to be applicable in different applicative contexts.

2.4.2 Base classifiers

Three classification algorithms have been chosen to yield the information sources for the classification tasks: decision trees, naive Bayesian and logistic regression. The differences in

Datasets	# samples	# features	# classes
Digits	1797	64	10
Waveform	5000	21	3
Wine	178	13	3
Cancer	801	20531	5
Segments	2310	19	7
CNAE	1080	856	9
MNIST	70000	784	10

Table 2.1 Dataset specifications

the classification philosophy for each algorithm is the reason behind our choice. Decision trees are learned using a non-parametric algorithm, logistic regression is a discriminative probabilistic classifier while naive Bayesian is a generative probabilistic classifier.

For each type of classifier, a homogeneous ensemble with parallel architecture of $K = 3$ classifiers is built, i.e. three decision trees, three logistic regressions or three naive Bayes classifiers but no mix of those. Appendix C gives the details of the chosen hyperparameter values and optimization techniques (when applicable) for each algorithm. Each member in the ensemble has identical initial settings however they do not see the same data and thus converge to different decision functions $\hat{c}_k$.

In short, the choices made for each algorithm consists in making them working moderately well as absolute performance values do not matter here. Indeed, relative performances among combination methods is what we seek as a fusion should work regardless of the parametrization quality.

2.4.3 Desirable experimental conditions to assess combination performances

Experiment design to compare classifier combination methods must take into account the following points:

- We need to avoid situations in which one of the base classifiers dominates the others in every aspects otherwise we should perform classifier selection, not fusion. Selection can be viewed as a special class of fusion but it can usually be achieved at a much lower computational cost.

- The base classifiers should preferably be weak, i.e. underfit the data. In this situation, fusion allows to visit a larger hypothesis set than those of base classifiers and we

can hope for improved accuracy. This is actually the basic idea of boosting which aggregates weak homogeneous classifiers in a sequential scheme to produce a strong ensemble. Ensemble methods can also mitigate overfitting but performance discrepancies will usually not have the same order of magnitude as in the underfitting case and thus it is harder to achieve statistical significance.

Some hyperparameter choices reflect our will to underfit. The maximum depth parameter of the classification trees is set to 3. Logistic regression is a linear model and naive Bayes relies on unrealistic class conditional independence assumptions which make these two prone to underfitting.

- The classifiers should learn from complementary pieces of information and yield diverse decision functions. Obviously, combining three nearly identical functions is pointless. We propose to train the base classifiers on mutually exclusive subsets of features to make them diverse enough. The next subsection details this procedure. Note that making them learn on different datasets does not imply a dramatically higher independence of their outputs as there are redundant information across features.

2.4.4 Implemented experimental protocol

As justified in the preceding paragraphs, it is necessary to (artificially) create diversity in classifier decisions so that the fusion process has the ability to bring an added value. Thus, after a random shuffle of the example entry indexes, we train the three base classifiers on three mutually exclusive sets of attributes in the feature space. Let φ denote a permutation on $\{1;\dots;d\}$ where d is the dimensionality of input space. Any example $\mathbf{x}^{(i)}$ is split into three smaller examples $\mathbf{x}^{(i,k)}$, $k \in \{1;2;3\}$ in this way

$$\mathbf{x}^{(i,1)} = \mathbf{x}^{(i)}_{\varphi(1):\varphi\left(\frac{d}{3}\right)}, \quad \mathbf{x}^{(i,2)} = \mathbf{x}^{(i)}_{\varphi\left(\frac{d}{3}+1\right):\varphi\left(\frac{2d}{3}\right)} \quad \text{and } \mathbf{x}^{(i,3)} = \mathbf{x}^{(i)}_{\varphi\left(\frac{2d}{3}+1\right):\varphi(d)}. \tag{2.18}$$

In the above equation, we assume that d is a multiple of $K = 3$ for simplicity. If not, the floor function should be used to obtain an integer. In other words, this experimental protocol is similar to the random space method [46] if we drew $\frac{d}{K}$ examples uniformly without replacement.

Splits are repeated over 100 iterations to allow reliable performance estimations. After each feature split, we wrap algorithm in a stratified 2 fold cross valiation (CV) loop (train/test split) so that half of the data points are used for performance evaluation only. Another level

of CV is applied to the train set to obtain $\mathscr{D}_{\text{train}}$ and $\mathscr{D}_{\text{val}}$ (see Algorithm 2). We use a 10 fold stratified CV for this second loop.

2.4.5 Combination method performances

The performances of the t-norm based possibilistic combination of classifiers is compared with the following benchmark of approaches :

- minimum rule based possibilistic combination,
- multiplicative rule based possibilistic combination,
- Bayes rule approach,
- weighted voting combination based on accuracies,
- meta classifier trained using stacking,
- classifier selection based on cross validated estimation of accuracies,
- maximally accurate individual classifier,
- centralized classifier trained on all training data.

Minimum and multiplicative rule approaches are equivalent to the t-norm based combination for specific values of λ. They are thus interesting to monitor to prove that adding one degree of liberty corresponding to the choice of λ is useful and that automatically tuning λ by cross validation works. The Bayes rule and the voting approaches are gold standards in the class of classifier ensemble methods. Since we are combining classifiers, a comparison with a learned fusion method is interesting. Thus, we examine the performances of stacking. Classifier selection allows us to show that, in general, fusion is more efficient than selection. We also provide the performances of the maximally accurate individual classifier, i.e. a classifier selection performed by an oracle which has access to the true performances (not estimations of these), and of the centralized classifier trained on all the data.

Results of classification trees, logistic regressions and naive Bayes classifiers are presented in tables 2.3, 2.4 and 2.5 respectively. For each experiment, the best accuracy is in bold characters. It is clear that the possibilistic t-norm combination outperforms the possibilistic minimum rule in the majority of the situations. Similarly, it outperforms stacking approach and the standard weighted vote very often. The t-norm possibilistic combination also achieves higher accuracy than classifier selection (except for the segments dataset when combining

naive Bayes base classifiers). The oracle classifier selection has more competitive results but is still outperformed by our approach in a majority of experiments. The oracle classifier selection is anyway just a reference and cannot be implemented in practice. However, the possibilistic t-norm combination method is always outperformed by the probabilistic Bayes rule and it achieved comparable results with the possibilistic product rule. This indicates that the proposed experimental protocol induces a low level of dependency between classifiers even when the combined base classifiers are homogeneous.

Classifiers	Poss. t-norm	Poss. mult.	Poss. min	Proba. Bayes	Stacking	Weighted Vote	Clf. Selection
Classification trees	2.71	2.28	4.42	1	6.14	4.85	5.57
Logistic regression	2.42	2.14	4.85	1.14	6.23	3.55	4.71
Naive Bayesian	3	2.28	5	1	6.57	4.71	5.71

Table 2.2 Average ranking of the combination methods across the seven datasets.

As expected, all combination approaches showed modest performances as compared to the centralized classifier because the latter has the advantage of observing all the training data in contrast to other methods that are only learned on limited fractions of the data. A striking exception to this conclusion can be observed for the decision tree ensembles. We believe this is explained by the drastic parametrization we chose (maximal depth $= 3$) which implies that the centralized decision tree underfits while ensembles can converge to more complex decision functions. In order to have a better view of the results and to draw strong conclusions, we present the rankings of all combination (and selection) approaches across all datasets for each classifier triplets. Obviously, the results of the table 2.2 demonstrate that the possibilistic t-norm combination has the third rank in average among all other combination methods (including classifier selection).

In addition, in many experiments, the classification accuracy discrepancies are greater than the corresponding standard deviations. Nevertheless, a statistical analysis is necessary to upraise their significance. We comment on this point in the next subsection. Also, each method classifies test samples in a meaningful way as compared to a random classifier. Some of the learned ensembles or base classifiers are nonetheless weak but remember that we have designed the experiments in this aim because the point here is to compare combination methods not to achieve state-of-the-art classification performances on each dataset.

In conclusion, these experiments demonstrate that the Bayes rule combination and the product rule outperform our proposed approach in the situation induced by the experimental protocol. We think that this protocol induces high independence levels −by training classifiers

on disjoint feature sets− for which a rule involving reinforcement is very suitable. To prove the interest of our method, the level of dependence in classifier decisions sould be increased. To do so, we decided to add to the ensemble of classifiers, a number of copies of a dummy classifier. This modification in the experimental protocol is actually very interesting because it allows to examine the robustness of the proposed method at two levels. The first one is the robustness to the dependency level because adding copies of the same classifier certainly increase dependency. The second one is the robustness to fault tolerance since dummy classifier decisions can be considered as noisy information. The results of these new experiments are provided in section 2.5.

Method	**Digits**	**Waveform**	**Cancer**	**CNAE**	**Segments**	**Wine**	**MNIST**
Poss. t-norm	67.85 $\pm$ 3.38	75.23 $\pm$ 1.08	90.76 $\pm$ 2.82	47.07 $\pm$ 4.38	73.63 $\pm$ 5.75	88.79 $\pm$ 4.04	62.62 $\pm$ 1.65
Poss. mult.	68.77 $\pm$ 3.13	75.31 $\pm$ 1.00	91.55 $\pm$ 2.85	46.06 $\pm$ 4.83	73.49 $\pm$ 5.48	92.65 $\pm$ 2.42	62.63 $\pm$ 1.58
Poss. min.	63.18 $\pm$ 3.43	63.86 $\pm$ 2.35	90.25 $\pm$ 2.68	45.43 $\pm$ 4.77	73.35 $\pm$ 5.59	86.17 $\pm$ 3.57	58.78 $\pm$ 1.16
Proba. Bayes	**72.01** $\pm$ 2.61	**75.51** $\pm$ 0.93	**92.40** $\pm$ 2.85	**56.56** $\pm$ 3.52	**75.49** $\pm$ 6.53	**93.34** $\pm$ 2.11	**64.59** $\pm$ 1.24
Stacking	34.21 $\pm$ 4.61	70.51 $\pm$ 1.56	85.20 $\pm$ 3.45	32.40 $\pm$ 5.98	59.82 $\pm$ 7.50	80.41 $\pm$ 8.95	37.20 $\pm$ 3.65
Weighted Vote	53.05 $\pm$ 3.62	75.08 $\pm$ 1.06	88.69 $\pm$ 0.57	30.92 $\pm$ 4.58	61.48 $\pm$ 6.07	92.35 $\pm$ 2.37	44.44 $\pm$ 1.65
Clf. Selection	47.97 $\pm$ 3.28	69.08 $\pm$ 1.21	86.72 $\pm$ 1.08	31.20 $\pm$ 2.19	61.73 $\pm$ 5.08	81.98 $\pm$ 6.16	43.37 $\pm$ 0.92
Max. Ind. Clf.	49.97 $\pm$ 2.61	70.10 $\pm$ 0.90	87.76 $\pm$ 1.01	33.27 $\pm$ 1.45	62.37 $\pm$ 4.69	89.94 $\pm$ 2.36	43.37 $\pm$ 0.92
Centralized Clf.	44.78 $\pm$ 2.57	70.89 $\pm$ 0.75	87.18 $\pm$ 0.66	35.02 $\pm$ 0.86	58.37 $\pm$ 2.80	89.49 $\pm$ 2.60	43.37 $\pm$ 0.92

Table 2.3 Average accuracies ($\pm$ standard deviations) of the combination of three decision trees with limited maximum depth (=3) over 100 iterations.

Method	**Digits**	**Waveform**	**Cancer**	**CNAE**	**Segments**	**Wine**	**MNIST**
Poss. t-norm	89.91 ± 1.27	83.24 ± 0.64	99.81 ± 0.06	84.08 ± 2.21	90.70 ± 1.23	88.85 ± 3.40	88.20 ± 0.07
Poss. mult.	91.32 ± 0.67	83.34 ± 0.60	99.81 ± 0.06	85.21 ± 1.42	90.68 ± 1.38	90.06 ± 3.21	86.46 ± 0.0005
Poss. min.	84.64 ± 1.09	71.59 ± 1.15	99.76 ± 0.05	76.43 ± 2.26	89.43 ± 1.25	87.84 ± 3.35	81.09± 0.0005
Proba. Bayes	91.46 ± 0.67	83.44 ± 0.57	99.81 ± 0.06	89.31 ± 1.20	91.15 ± 1.23	91.47 ± 1.84	87.46 ± 0.0
Stacking	31.33 ± 3.88	74.24 ± 3.41	61.95 ± 6.26	28.35 ± 3.56	49.62 ± 4.88	59.83 ± 9.14	48.21 ±0.009
Weighted Vote	89.46 ± 0.88	83.26 ± 0.69	99.81 ± 0.06	79.11 ± 2.20	86.02 ± 2.60	88.45 ± 3.54	77.54 ± 0.0003
Clf. Selection	85.89 ± 2.20	78.09 ± 1.59	99.81 ± 0.07	72.15 ± 3.79	85.83 ± 3.76	85.72 ± 4.73	84.68 ± 0.0005
Max. Ind. Clf.	86.85 ± 1.63	78.70 ± 1.29	**99.84** ± 0.07	73.86 ± 3.05	86.26 ± 3.42	88.96 ± 2.55	84.68 ± 0.0006
Centralized Clf.	**95.12** ± 0.44	**86.52** ± 0.23	99.82 ± 0.06	**93.47** ± 0.52	**92.20** ± 0.27	**94.85** ± 1.19	**90.75** ± 0.0002

Table 2.4 Average accuracies (± standard deviations) of the combination of three logistic regression classifiers 100 iterations.

Method	**Digits**	**Waveform**	**Cancer**	**CNAE**	**Segments**	**Wine**	**MNIST**
Poss. t-norm	79.13 ± 2.47	79.59 ± 0.81	83.46 ± 2.00	78.93 ± 2.81	84.21 ± 2.13	92.10 ± 2.71	64.64 ± 1.11
Poss. mult.	79.90 ± 2.45	79.91 ± 0.76	84.59 ± 1.43	80.39 ± 1.82	83.57 ± 2.47	94.80 ± 1.86	64.55 ± 1.08
Poss. min.	74.17 ± 2.62	70.98 ± 2.04	78.78 ± 2.06	70.80 ± 2.39	82.81 ± 2.33	90.66 ± 2.73	60.37 ± 1.08
Proba. Bayes	82.10 ± 1.93	80.00 ± 0.80	**88.52** ± 1.42	86.23 ± 1.36	**85.31** ± 1.92	95.13 ± 1.61	**66.72** ± 1.10
Stacking	59.57 ± 5.41	78.16 ± 1.08	73.77 ± 4.97	61.52 ± 4.90	75.77 ± 4.07	86.58 ± 9.30	43.88 ± 1.89
Weighted Vote	69.36 ± 3.21	78.51 ± 0.74	83.98 ± 1.37	70.04 ± 2.20	73.11 ± 1.93	94.78 ± 1.77	46.90 ± 1.06
Clf. Selection	67.43 ± 5.23	75.88 ± 1.82	77.57 ± 2.42	65.33 ± 3.68	77.26 ± 4.35	89.92 ± 3.47	50.67 ± 2.02
Max. Ind. Clf.	68.35 ± 4.40	76.57 ± 1.37	79.60 ± 1.46	66.83 ± 2.70	77.83 ± 3.97	93.29 ± 1.73	50.67 ± 2.02
Centralized Clf.	**83.43** ± 1.24	**80.93** ± 0.10	81.62 ± 2.01	**90.02** ± 0.99	79.71 ± 0.50	**96.90** ± 1.08	50.67 ± 2.02

Table 2.5 Average accuracies (± standard deviations) of the combination of three naive Bayesian classifiers with Gaussian class conditional distribution over 100 iterations.

2.4.6 Statistical validation

To draw reliable conclusions based on the experiments presented in the previous subsection, we use a statistical test to validate the significance of performance discrepancies between

the t-norm fusion method and concurrent approaches (multiplicative, minimum, Bayes rules, stacking, weighted voting, selected and maximally accurate base classifiers and the centralized classifier). To that end, we apply a Wilcoxon signed-rank test.

The Wilcoxon signed-rank test scope is to compare repeated measurements of two variables in order to decide if there is a statistically significant difference in the two series of measurements. It is a non parametric hypothesis test in which differences between pairs of values are computed and interpreted according to two pre-defined hypotheses (null and alternative hypotheses denoted by H_0 and H_1). The outputs of the test are a statistic (roughly speaking the sum of the ranks) and a p-value. The two series of measurements in our case are the series of accuracies of two approaches. The detailed tables are presented in Appendix B. Each table contains 5 columns: name of the dataset, the value of the statistic, the p-value, the conclusion of the test (i.e. passed or not) and the outperforming classification method if the test is passed. Note that we considered a threshold of 5% to reject the null hypothesis, i.e. if p-value ≤ 0.05. Having a p-value below this threshold means that the examined series of accuracies is highly unlikely under the null hypothesis which is thus rejected. The null hypothesis in this test states that the distribution of accuracy signed differences is symmetric.

Pairs of methods	# ✓	# ✗	#t-norm wins	#t-norm loses	#tie
t-norm *vs.* Mult.	16	5	3	12	1
t-norm *vs.* Min	21	0	21	0	0
t-norm *vs.* Bayes	21	0	0	20	1
t-norm *vs.* Stacking	21	0	21	0	0
t-norm *vs.* Weighted vote	20	1	16	3	1
t-norm *vs.* Clf. Selection	21	0	21	0	0
t-norm *vs.* Max. Ind. Clf.	20	1	17	3	0
t-norm *vs.* Centralized Clf.	19	2	9	10	0

Table 2.6 Synthetic summary of Wilcoxon tests. Detailed tables are found in B.

Table 2.6 summarizes pairwise comparisons between the t-*norm* based possibilistic approach to each concurrent one. The Wilcoxon signed-rank test shows that the results of the previous subsection are statistically validated since in the majority of the situations the test is passed. The t-norm possibilistic rule achieves better accuracies than many approaches except for Bayes combination. The t-norm also showed comparable results to the possibilistic product rule. Thus, the conclusions of the previous subsection are confirmed.

2.5 Classifier fault tolerance study

In order to check the tolerance of the combination to noisy information, we carry out several experiments in which we inject unreliable information sources. For each experiment, we kept the previous base classifier pool, i.e. decision trees, naive Bayes classifiers and logistic regressions, but we added a number of dummy classifiers[1] to the classifiers pool. We began by adding one dummy classifier to the three base classifiers and progressively add up to 25 copies of this dummy classifier.

Rankings of each method, for each situation, are presented in tables 2.7, 2.8 and 2.9. The performances of the possibilistic multiplicative and the probabilistic Bayes rules are degraded as the amount of noisy information increases, i.e. adding $\{1,..,25\}$ dummy classifiers. This is probably due to the fact that these two combination rules are based on «multiplicative type of operations» which is a non-idempotent operation and reinforces the impact of irrelevant predictions each time a new dummy classifier is added. In contrast, the possibilistic minimum rule obtains better ranks when we increase the number of dummy classifiers. This is compatible with the previous conclusion since the minimum rule is an idempotent operation which takes into consideration only once the same noisy information regardless how many copies of the dummy classifier are added. As expected, as we increased the number of dummy classifiers the weighted vote shows very poor performances as compared to other approaches. The t-norm possibilistic combination achieves better performances than the Bayes rule and the multiplicative possibilistic rules in case of high amount of noise while it achieves better performances than the minimum rule in case of low amount of noise. This is due to its flexibility and ability to adapt itself to different dependency levels between the classifiers thanks to the parameter λ. In addition, the variation in the ranking of the t-norm approach across the lines in tables 2.7, 2.8 and 2.9 is smoother the Bayes and possibilistic minimum approaches which seems to be a form of compromise and robustness. Observe that when a large number of dummy classifiers is added, the minimum possibilistic rule outperforms the t-norm approach which tends to show that the estimation of λ is suboptimal in these experiments.

It is also obvious that the t-norm is better ranked than the stacking approach for the three different classifier ensembles. However, the classifier selection becomes more competitive than other combination approaches, as the amount of noisy information increases. This is

[1] A dummy classifier assigns all test samples to the most frequent class in the training set. We used the implementation from the python Scikit-learn library 0.17.1 `sklearn.dummy.DummyClassifier()`

due to the fact that the selection process discards completely noisy information in contrast to combination methods that take them into consideration.

In order to get better insights of the results, we present the variation of accuracies for each dataset for the three classifier ensembles (decision trees, logistic regressions and the naive Bayesian classifiers). For the sake of clarity, we provide the accuracies of the most competitive methods among all approaches according to the ranking tables, thus stacking and weighted vote are excluded from these figures. Accuracies are presented in Figure 2.3 where the $x-$axis represents the number of added dummy classifiers and the $y-$axis represents the accuracy. The legend of the figures are in the caption.

By observing the variation of accuracies as the amount of noise increases, we notice that the possibilistic product rule and the probabilistic Bayes rules have the largest variations while the t-norm possibilistic rule, the minimum rule and the classifier selection manage themselves to be more robust.

Classifier fault tolerance experiments demonstrate that the proposed approach obtains some of the best accuracies among other combination methods when noisy information sources are injected in classifier decisions (except for the MNIST dataset). In addition, it has been outlined that a classifier selection might be more advantageous than a combination technique when there is a high level of noise and also that a better estimation method of the parameter λ could be envisaged. Actually, the estimated accuracies used in classifier selection could be used to discard irrelevant classifiers before performing classifier fusion.

In conclusion, the experiments of section 2.4.5 showed that multiplication-based rules (Bayes rule and possibilistic product rule) achieve higher performances than the possibilistic t-norm rule due to the low level of dependency induced by the experimental protocol. However, the modification of the protocol $-$by adding dummy classifiers$-$ proved that the t-norm possibilistic rule achieves more stable performances than all other combination rules (including classifier selection). This allows us to deduce that the proposed approach is suitable to combine dependent classifiers thanks to its parameter λ and also tolerant to the presence of noisy classifiers. Thus, the major benefit of the t-norm possibilistic rule is the robustness which was previously claimed in the manuscript.

Classifiers	Poss. t-norm	Poss. mult.	Poss. min	Proba. Bayes	Stacking	Weighted Vote	Clf. Selection
Base + 1 dummy	2.43	2.86	4.43	1.0	6.43	5.29	5.57
Base + 2 dummies	2.86	3.86	3.71	1.0	5.86	6.43	4.29
Base + 3 dummies	3.14	4.71	2.86	1.0	5.14	7.0	4.14
Base + 4 dummies	2.86	5.14	2.57	1.43	5.0	7.0	4.0
Base + 5 dummies	3.29	5.86	2.43	1.57	4.29	7.0	3.57
Base + 6 dummies	3.0	5.86	2.43	2.0	4.14	7.0	3.57
Base + 7 dummies	2.86	5.86	2.29	2.29	4.14	7.0	3.57
Base + 8 dummies	2.86	6.0	2.29	2.43	4.0	7.0	3.43
Base + 9 dummies	3.0	6.0	2.29	2.57	4.0	7.0	3.14
Base + 10 dummies	2.86	6.0	2.29	2.57	4.0	7.0	3.29
Base + 11 dummies	2.43	5.14	2.14	2.43	3.29	6.0	2.57
Base + 12 dummies	2.57	5.14	2.14	2.57	3.14	6.0	2.43
Base + 13 dummies	2.57	5.14	2.14	2.57	3.14	6.0	2.43
Base + 14 dummies	2.43	5.14	2.0	2.86	2.86	6.0	2.71
Base + 15 dummies	2.29	5.14	2.0	3.14	2.86	6.0	2.57
Base + 16 dummies	2.57	5.14	1.71	3.0	2.86	6.0	2.71
Base + 17 dummies	2.71	5.14	2.14	2.86	2.71	6.0	2.43
Base + 18 dummies	2.43	5.14	2.0	3.0	2.71	6.0	2.71
Base + 19 dummies	2.71	5.14	2.43	3.0	2.57	6.0	2.14
Base + 20 dummies	2.71	5.14	2.0	3.14	2.71	6.0	2.29
Base + 21 dummies	2.71	5.29	2.14	3.14	2.71	5.86	2.14
Base + 22 dummies	2.71	5.29	2.0	3.14	2.71	5.86	2.29
Base + 23 dummies	2.71	5.29	2.0	3.14	2.71	5.86	2.29
Base + 24 dummies	2.57	5.29	1.86	3.43	2.71	5.86	2.29
Base + 25 dummies	2.43	5.29	2.0	3.29	2.71	5.86	2.43

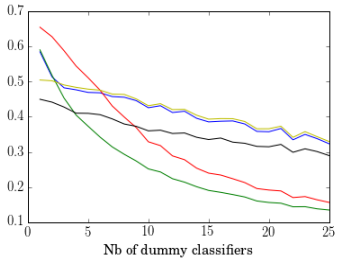
Table 2.7 Average ranking of the combination methods across the seven datasets. The base classifiers are a triplet of classification trees to which dummy classifiers are added.

Classifiers	Poss. *t*-norm	Poss. mult.	Poss. min	Proba. Bayes	Stacking	Weighted Vote	Clf. Selection
Base + 1 dummy	2.86	2.86	5.71	1.14	6.71	4.14	4.57
Base + 2 dummies	3.43	3.71	5.0	1.14	6.43	5.43	2.86
Base + 3 dummies	3.71	4.86	3.71	1.14	5.43	7.0	2.14
Base + 4 dummies	3.43	5.14	3.43	1.29	5.43	7.0	2.29
Base + 5 dummies	3.29	5.29	3.29	1.57	5.29	7.0	2.29
Base + 6 dummies	3.57	5.43	3.14	1.57	5.0	7.0	2.29
Base + 7 dummies	3.57	5.43	2.86	1.86	5.0	7.0	2.29
Base + 8 dummies	3.43	5.57	2.71	2.0	4.86	7.0	2.43
Base + 9 dummies	3.29	5.57	2.71	2.14	4.86	7.0	2.43
Base + 10 dummies	3.43	5.57	2.57	2.0	4.86	7.0	2.57
Base + 11 dummies	2.57	4.86	2.57	2.14	3.86	6.0	2.0
Base + 12 dummies	2.71	4.86	2.43	2.14	3.71	6.0	2.14
Base + 13 dummies	2.57	4.86	2.29	2.29	3.57	6.0	2.43
Base + 14 dummies	2.57	4.86	2.29	2.29	3.57	6.0	2.43
Base + 15 dummies	2.57	4.86	2.29	2.29	3.57	6.0	2.43
Base + 16 dummies	2.71	5.0	2.14	2.29	3.43	6.0	2.43
Base + 17 dummies	2.57	5.0	2.14	2.43	3.43	6.0	2.43
Base + 18 dummies	2.57	5.0	2.0	2.57	3.43	6.0	2.43
Base + 19 dummies	2.57	5.0	2.43	2.57	3.43	6.0	2.0
Base + 20 dummies	2.43	5.14	2.43	2.71	3.43	5.86	2.0
Base + 21 dummies	2.43	5.14	2.43	2.71	3.43	5.86	2.0
Base + 22 dummies	2.43	5.14	2.43	2.86	3.29	5.86	2.0
Base + 23 dummies	2.43	5.14	2.43	2.86	3.29	5.86	2.0
Base + 24 dummies	2.29	5.14	2.29	3.14	3.29	5.86	2.0
Base + 25 dummies	2.71	5.14	2.29	2.71	3.29	5.86	2.0

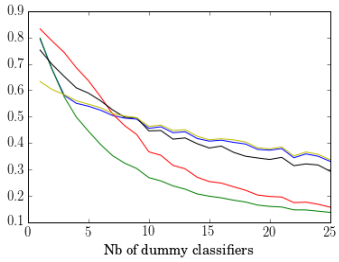
Table 2.8 Average ranking of the combination methods across the seven datasets. The base classifiers are a triplet of logistic regressions to which dummy classifiers are added.

Classifiers	Poss. *t*-norm	Poss. mult.	Poss. min	Proba. Bayes	Stacking	Weighted Vote	Clf. Selection
Base + 1 dummy	2.71	3.14	5.86	1.0	6.0	4.86	4.43
Base + 2 dummy	3.57	3.57	4.57	1.0	5.57	6.71	3.0
Base + 3 dummy	3.43	5.29	3.43	1.14	5.14	7.0	2.57
Base + 4 dummy	3.71	5.43	3.14	1.29	4.86	7.0	2.57
Base + 5 dummy	3.29	5.57	2.86	1.43	5.0	7.0	2.86
Base + 6 dummy	3.43	5.57	2.71	1.43	5.0	7.0	2.86
Base + 7 dummy	3.29	5.57	2.43	2.0	5.0	7.0	2.71
Base + 8 dummy	3.29	5.71	2.43	2.0	4.86	7.0	2.71
Base + 9 dummy	3.29	5.86	2.43	2.0	4.86	7.0	2.57
Base + 10 dummy	3.29	6.0	2.43	2.0	4.71	7.0	2.57
Base + 11 dummy	2.71	5.0	2.29	1.86	4.29	6.0	1.86
Base + 12 dummy	2.71	5.0	2.14	2.0	4.43	6.0	1.71
Base + 13 dummy	2.43	5.0	1.86	2.29	4.43	6.0	2.0
Base + 14 dummy	2.57	5.14	1.71	2.29	4.29	6.0	2.0
Base + 15 dummy	2.57	5.0	1.71	2.29	4.43	6.0	2.0
Base + 16 dummy	2.57	5.0	1.71	2.29	4.43	6.0	2.0
Base + 17 dummy	2.43	5.0	1.86	2.29	4.43	6.0	2.0
Base + 18 dummy	2.29	5.14	1.71	2.57	4.43	5.86	2.0
Base + 19 dummy	2.57	5.14	2.0	2.57	4.29	5.86	1.57
Base + 20 dummy	2.57	5.14	2.0	2.57	4.29	5.86	1.57
Base + 21 dummy	2.43	5.14	1.86	2.86	4.29	5.86	1.57
Base + 22 dummy	2.43	5.14	2.14	2.57	4.29	5.86	1.57
Base + 23 dummy	2.57	5.14	1.86	2.71	4.29	5.86	1.57
Base + 24 dummy	2.29	5.14	1.86	3.0	4.29	5.86	1.57
Base + 25 dummy	2.43	5.14	1.71	3.0	4.29	5.86	1.57

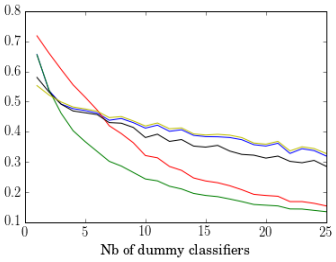
Table 2.9 Average ranking of the combination methods across the seven datasets. The base classifiers are a triplet of naive Bayesian to which dummy classifiers are added.



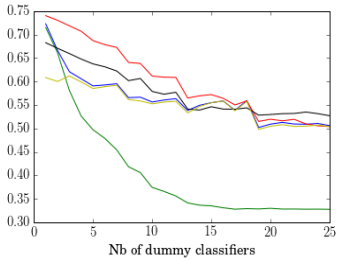
(a) Digits (decision trees)



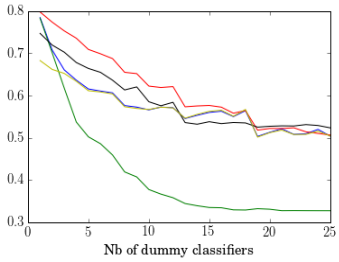
(b) Digits (log. reg.)



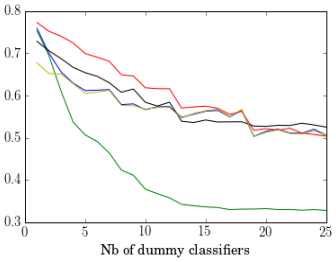
(c) Digits (naive Bayesian)



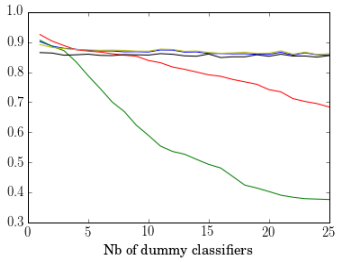
(d) Waveform (decision trees)



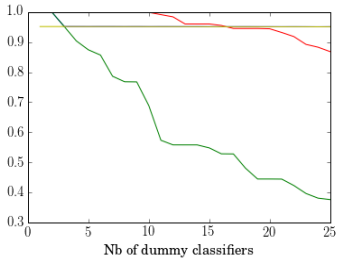
(e) Waveform (log. reg.)



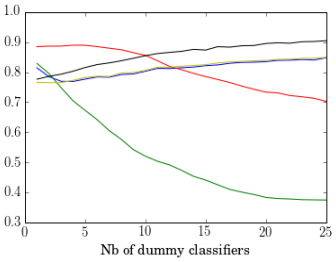
(f) Waveform (naive Bayesian)



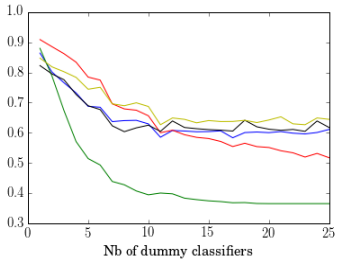
(g) Cancer (decision trees)



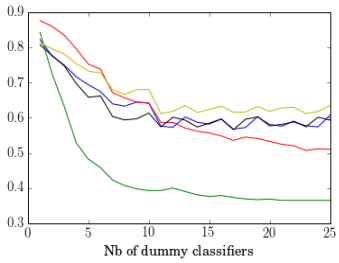
(h) Cancer (log.reg.)



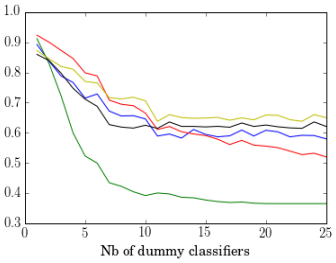
(i) Cancer (naive Bayesian)



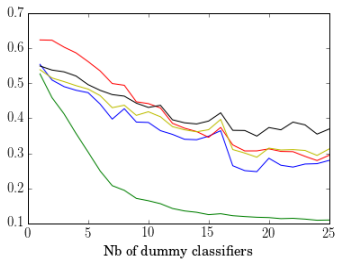
(j) Wine (decision trees)



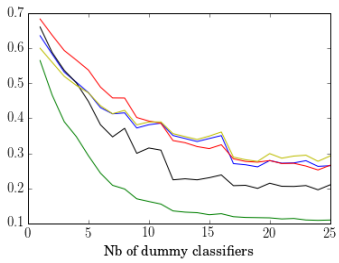
(k) Wine (log. reg.)



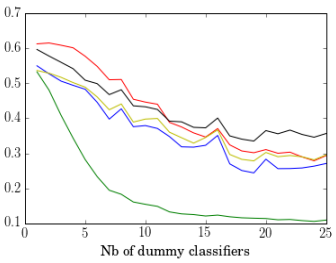
(l) Wine (naive Bayesian)



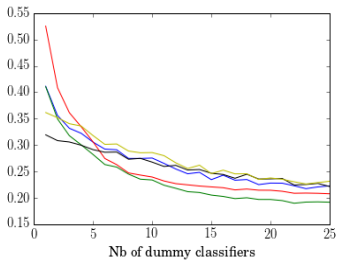
(m) Segments (decision trees)



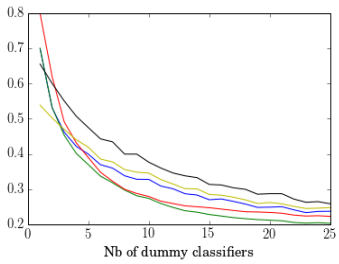
(n) Segments (log. reg.)



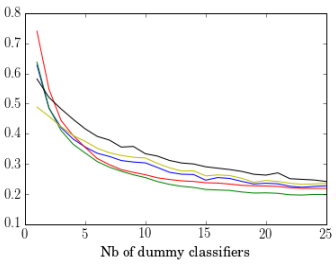
(o) Segments (naive Bayesian)



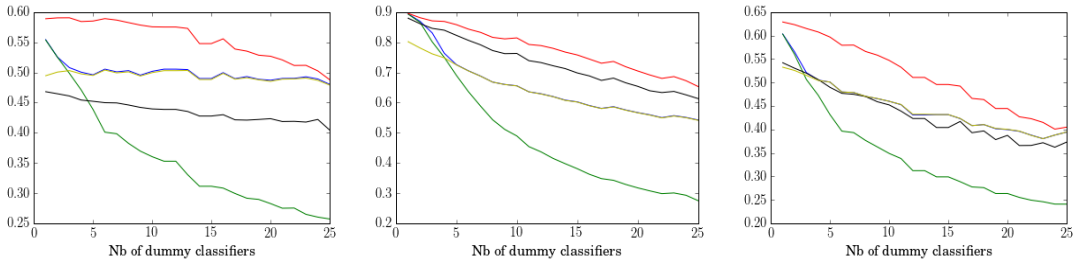
(p) CNAE (decision trees)



(q) CNAE (log. reg.)



(r) CNAE (naive Bayesian)



(a) MNIST (decision trees) (b) MNIST (log. reg.) (c) MNIST (naive Bayesian)

Fig. 2.3 Global accuracies for all datasets. The t-norm rule (—), Bayes rule (—), possibilistic product rule (—), minimum rule (—), classifier selection (—).

2.6 Conclusion

This chapter presents the first original contribution of this Ph.D. We propose a new classifier combination strategy based on the estimated performances of the base classifiers. The estimated performances are conditional probabilities of the true class label given the label prediction issued by a base classifier. They are obtained via cross validation and they are derived from confusion matrices. In order to make optimal decision on the true class label given all the classifier predictions, one needs to maximize the probability of the true class label given the predicted ones. This maximization is not possible with limited complexity unless one makes unrealistic conditional independence assumptions. In this approach, we propose to circumvent independence assumptions by transposing the combination problem in the possibility theory framework. In this framework, we can apply parametrized t-*norm* rules to aggregate information contained in the confusion matrices. The proposed method is flexible regarding two important aspects: the parameter of the t-*norm* allows a variety of treatments of dependence issues and there is no pre-requisite on the base classifiers constituting the ensemble.

The method was tested on 7 datasets with the Aczel-Alsina t-*norm* family and homogeneous ensembles of three classifiers. Several supervised learning algorithms have been tested to train the base classifiers. The results report a significant accuracy increment for our possibilistic approach to dependent classifier combination as compared to 3 other combination methods (possibilistic minimum rules, weighted vote and stacking) and to classifier selection. Our approach achieves close levels of performances as compared to the possibilistic multiplicative rule and is outperformed by the probabilistic Bayes rule approach because the information delivered by the base classifiers is not very redundant in this experiment. The experiments also show that the t-*norm* parameter can be efficiently tuned by cross validation, i.e. without human user supervision.

A classifier fault tolerance study was also carried out in this chapter to check the sensitivity of each method to noisy information. We kept the panel of classifiers as in the aforementioned experiments, i.e. triplets of decision trees, naive Bayesian and logistic regression classifiers, but we added a number of dummy classifiers to the pool. The results show that the possibilistic *t*-norm combination achieves better robustness among the approaches in the benchmark. This robustness lies in the ability of the proposed method to adapt itself to the level of dependency between classifiers and cope with poorly accurate ones.

In the following chapter, we introduce another combination approach that relies on copulas which are functions to handle statistical dependency between variables. In this

approach, we use a parametrized copula to model the dependency between classifiers. We keep following the first principle evoked in the introduction of this chapter which is the usage of contextual information in the fusion schemes. In contrast to this chapter, our next contribution with respect to dependent classifier combination remains in the probabilistic framework.

Chapter 3

Probabilistic copula based combination approach

3.1 Introduction

In this thesis, we place ourselves in a decentralized learning context where we aim at introducing scalable fusion schemes achieving better performances in terms of generalization error or robustness as compared to other common combination approaches as well as to classifier selection approaches.

Following the analysis of the state of the art in classifier combination that we carried out in chapter 1, we enumerate two requirements:

(i) the usage of contextual information as basic ingredient in the fusion process,

(ii) the ability of the proposed method to adapt itself to the statistical dependence between classifiers.

In contrast to the t-norm combination approach which is introduced in chapter 2 and formalized using the possibilistic framework, we propose in this chapter a probabilistic model of classifier aggregation.

In this probabilistic approach, each base classifier is first trained on a fraction of the training dataset. These fractions may or may not overlap and no assumption is made in this regard. In a second time, we select a probabilistic classification rule relying on the base classifier predictions. We investigate a model relying on conditional probabilities of classifier outputs given the true class of an input. These distributions can be used as building blocks to classify unseen examples as those maximizing class probabilities given all classifier outputs

[18, 57]. The originality of our approach consists in resorting to copula functions to obtain a relatively simple model of joint conditional distributions of the base classifier outputs given the true class.

The next section gives a formal presentation of the classifier combination probabilistic model studied in this chapter. We also provide elementary background on copulas. In Section 3.3.2, our new probabilistic classifier aggregation model relying on Gaussian copulas is introduced. The performance of this approach in both classification accuracy and robustness is assessed in section 3.5 where we carry several numerical experiments on both synthetic and real data sets. Section 3.5.4 presents a similar statistical validation as the one of chapter 2 and finally a conclusion entails the chapter.

3.2 Combination problem statement

We address an aggregation problem which consists in predicting the class label of an example $\mathbf{x}$ from the set of predictions $\hat{c}_1(\mathbf{x}),\ldots,\hat{c}_K(\mathbf{x})$ issued by K base classifiers. Under 0-1 loss, the optimal decision rule is

$$\hat{c}(\mathbf{x}) = \underset{y\in\Omega}{\arg\max}\ \ p(y|\hat{c}_1(\mathbf{x}),\ldots,\hat{c}_K(\mathbf{x})), \tag{3.1}$$

$$= \underset{y\in\Omega}{\arg\max}\ \ p(\hat{c}_1(\mathbf{x}),\ldots,\hat{c}_K(\mathbf{x})|y)\times p(y). \tag{3.2}$$

As said previously, an estimation of the joint distributions $p(\hat{c}_1(\mathbf{x}),\ldots,\hat{c}_K(\mathbf{x})|y)$ (for $y\in\Omega$) on a validation set is not a good choice because the dimensionality of the parameter space grows exponentially with K (the number of base classifiers). Since the size of a validation set is usually limited, then we would not be able to avoid overfitting. A simple solution to circumvent this question is to make conditional independence assumptions in which case linear complexity in K can be achieved:

$$\hat{c}(\mathbf{x}) = \underset{y\in\Omega}{\arg\max}\ \ p(y)\times\prod_{i=1}^{K} p(\hat{c}_i(\mathbf{x})|y). \tag{3.3}$$

We remind the readers that the above classification rule is referred to as the «Bayes rule approach» throughout this manuscript. In spite of the unrealistic aspect of the independence assumptions allowing to derive this approach, we will see that it achieves nice classification accuracy on several occasions. We believe this is explained by the same reason as the one

behind naive Bayes classifier efficiency. This model is an efficient technique although it also relies on unrealistic independence assumptions. Indeed, the inadequacy of these assumptions is compensated by a dramatic reduction of the number of parameters to learn making the technique less prone to overfitting.

The Bayes rule approach takes into account the individual performances of the base classifiers which are featured by the probabilities $p(\hat{c}_i(\mathbf{x})|y)$ and this already fulfills one of the fundamental requirements that we have stated in the introduction of this chapter. The second requirement is to obtain a model that adapt itself to the level of dependency between base classifiers. The Bayes rule approach fails to possess this flexibility. In the review of the state-of-the-art that we carried out in chapter 1, we mentioned a work by Kim and Ghaharamani [57] in which they propose to infer the parameters of each distribution involved in (3.3) using a hierarchical Bayesian model. Moreover, they also introduce two extended models that take into account classifier dependencies. The first idea they develop is to introduce hidden Bernoulli variables depicting the difficulty to classify each training data point. If the point is hard to classify, the model is the same as before otherwise confusion matrices are tied across classifiers. In this latter case, the probabilities $p(\hat{c}_i(\mathbf{x})|y)$ are not learned from data but instead they are set to

$$p(\hat{c}_i(\mathbf{x})|y) = \begin{cases} \varepsilon & \text{if } \hat{c}_i(\mathbf{x}) = y \\ 1-\varepsilon & \text{otherwise} \end{cases}, \tag{3.4}$$

where ε is a hyperparameter living in $(0;1]$ and for all $i \in \{1,\ldots,K\}$. Although this idea drifts apart from the independence assumptions, it is more a semi-local model in the input space than a model with an explicit component addressing the classifier dependency question. Such an explicit solution is proposed in their second idea which consists in a Markov random field in which each pair of classifier is a clique of the graphical model. In any of these models, one must resort to MCMC and rejection sampling in order to make the solution operational. We argue that the dependency can be efficiently taken into using simpler models with better computational tractability.

In this chapter, we intend to derive such a model for the joint conditional distributions $p(\hat{c}_1(\mathbf{x}),\hat{c}_2(\mathbf{x}),..,\hat{c}_K(\mathbf{x})|y)$. We propose to decompose each of these distributions using a copula function $\mathscr{C}$ which is a cumulative distribution with uniform marginals. In a nutshell, this decomposition allows to separate dependency information embodied by the copula function from individual classifier information (captured by the marginals). The growing popularity of copula functions stems from Sklar's theorem which asserts that, for every

random vector $\mathbf{L} \sim f$, there exist a copula $\mathscr{C}$ such that $F = \mathscr{C} \circ \mathbf{G}$ where F is the cumulative version of distribution f and $\mathbf{G}$ is a vector whose entries are the cumulative marginals $G_k(a) = F(\infty, \ldots, \infty, a, \infty, \ldots, \infty)$ for any a in the k-dimensional domain of f. However, for some reasons that will be explained later in section 3.3.2, we will use another version of this theorem that is based on the density c of the copula $\mathscr{C}$. Using this version of Sklar's theorem, we obtain directly the density of the joint conditional probability distribution instead of its cumulative version.

The proposed method is presented in details (how to compute the joint distributions) in the next section.

3.3 Copula based probabilistic combination of classifiers

In this section, we give a detailed presentation of the proposed probabilistic copula-based approach to dependent classifier combination. As previously mentioned, suitable copula functions for describing random variable dependencies will be used to aggregate classifier decisions based on their estimated performances. We first recall the basic background of copula functions and then we present the combination process.

3.3.1 Basic background about copulas

In this subsection, some necessary copula related notions are introduced. We first give some notations. Let $(x,y) \in \mathbb{R}^2$ denote of pair of real variables and let $\mathbb{D}_x = [x_1; x_2]$ and $\mathbb{D}_y = [y_1; y_2]$ denote two closed intervals to which belong x and y respectively.

2−**increasing function**

The Cartesian product $\mathbb{D}_x \times \mathbb{D}_y = [x_1; x_2] \times [y_1; y_2]$ defines a rectangle $\mathbf{A}$ in $\mathbb{R}^2$ whose vertices are (x_1, y_1), (x_1, y_2), (x_2, y_1) and (x_2, y_2). If the domain of a function F is a subset of $\mathbb{R}^2$ and its co-domain is a subset of $\mathbb{R}$, then F is called a *2-place real* function.

Definition 3.3.1. Let F denote a *2-place* real function and $\mathbf{A}$ the rectangle whose vertices are defined by the Cartesian product $\mathbb{D}_x \times \mathbb{D}_y$. The *F-volume* (Figure 3.1) of $\mathbf{A}$ is computed as follow:

$$V_F(\mathbf{A}) = F(x_1, y_1) + F(x_2, y_2) - F(x_1, y_2) - F(x_2, y_1). \tag{3.5}$$

Moreover, F is *2-increasing* if $V_F(\mathbf{A}) \geq 0$ for all rectangles $\mathbf{A}$ belonging to the domain of F.

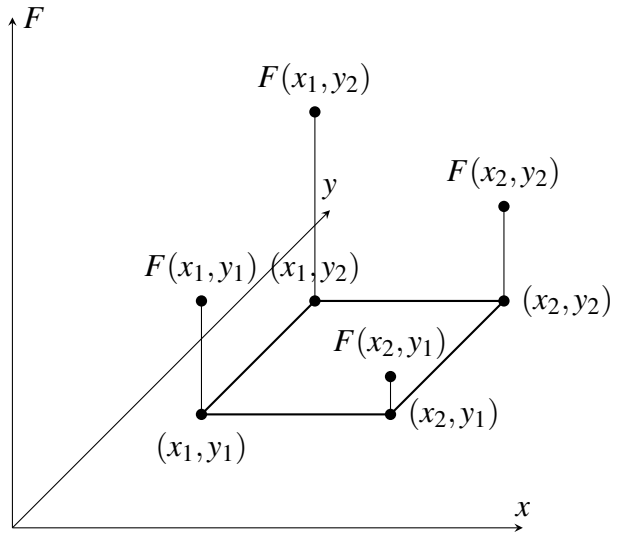


Fig. 3.1 A 3d view of *F-volume* computation of the rectangle in the horizontal plane.

Grounded function

Suppose that the domain of some function F is a subset of $\mathbb{R}^2$ defined as the Cartesian product of two closed interval: $\mathbb{D}_x \times \mathbb{D}_y = [x_1;x_2] \times [y_1;y_2]$, then F is *grounded* iff $F(x,y_1) = F(x_1,y) = 0$ for any $(x,y) \in \mathbb{D}_x \times \mathbb{D}_y$.

Margins

Let x_2 and y_2 denote the upper bounds of two closed intervals $\mathbb{D}_x$ and $\mathbb{D}_y$ whose Cartesian product is the domain of some function F. The function F has two *margins* that are respectively functions Marg_1 and Marg_2 given by:

- $\text{Marg}_1(x) = F(x,y_2)$ for any $x \in \mathbb{D}_x$,
- $\text{Marg}_2(y) = F(x_2,y)$ for any $y \in \mathbb{D}_y$.

Copulas

Building upon the aforementioned concepts, we can now introduce first sub-copulas which are *2-increasing*, *grounded* functions with specific *margins*.

Definition 3.3.2. Considering a bivariate function $\mathscr{C}' : \mathbb{D}_x \times \mathbb{D}_y \longrightarrow \mathbb{D}$, *s.t* $\mathbb{D}_x \subseteq \mathbb{D}$, $\mathbb{D}_y \subseteq \mathbb{D}$ and $\{0,1\} \subset \mathbb{D}_x, \mathbb{D}_y$, then $\mathscr{C}'$ is a (2 dimensional) sub-copula if:

1. $\mathscr{C}'$ is 2-increasing,

2. $\mathscr{C}'$ is grounded,

3. $\forall(x,y) \in \mathbb{D}_x \times \mathbb{D}_y$, $\mathscr{C}'(x,1) = x$ and $\mathscr{C}'(1,y) = y$.

Eventually, a copula is formally defined as follows:

Definition 3.3.3 (Analytical definition)**.** Considering a (2 dimensional) sub-copula $\mathscr{C} : \mathbb{D}_x \times \mathbb{D}_y \longrightarrow \mathbb{D}$, the function $\mathscr{C}$ is a (2 dimensional) copula iff $\mathbb{D}_x = \mathbb{D}_y = \mathbb{D}$.

The above definition is the analytical version of the definition of copulas and it does not relate to probability theory. However, copulas are almost essentially used within this framework. Indeed, in 1959, Sklar [90] introduced copulas for the first time to model the dependence between random variables from the joint probability distribution. Copulas are functions that offer a way to relate marginal distributions to the multivariate distribution. The probabilistic version of the definition of copulas is the following

Definition 3.3.4 (Probabilistic definition)**.** A (K dimensional) copula $\mathscr{C}$ is the restriction to $[0;1]^K$ of the multivariate cdf[1] of a random vector $\mathbf{U}$, called a copula representer, whose marginals are uniformly distributed on $[0;1]$.

The analytical version of the definition of copulas is easily generalized to the K dimensional case. We can now state Sklar's theorem:

Theorem 1 (Sklar's theorem)**.** *Suppose F is a bivariate (multivariate in a general case) cdf of the pair of random variables (X,Y) and G_1 and G_2 are the marginal cdfs of X and Y respectively, then there is a copula function $\mathscr{C}$ satisfying the following equality,*

$$F(x,y) = \mathscr{C}(G_1(x), G_2(y)), \tag{3.6}$$

for any (x,y) in the domain of F.

Reciprocally, if $\mathscr{C}$ is a copula and G_1 and G_2 are the univariate cdfs of X and Y, then F is the associated joint cdf for the pair (X,Y).

A corollary derived from theorem 1 is given as follows:

Corollary 3.3.1 (Corollary of Sklar's theorem)**.** Let F, G_1, G_2 and $\mathscr{C}$ denote the same functions as defined in theorem 1, and suppose G_1 and G_2 are continuous, then, for any $(x,y) \in [0;1]^2$, we have,

$$\mathscr{C}(x,y) = F(G_1^{-1}(x), G_2^{-1}(y)) \tag{3.7}$$

[1] cumulative distribution function.

where $G_1^{-1}(x) = \inf\{u \text{ s.t. } G_1(u) \leq x\}$ and $G_2^{-1}(y) = \inf\{v \text{ s.t. } G_2(v) \leq y\}$ are the quantile functions.

The corollary is an important result in practice since it allows to construct a dependency model between two variables knowing their joint and marginal cumulative distribution functions. We also deduce that, in the continuous case, there is only one copula satisfying Sklar's theorem.

There exist also a version of Sklar's theorem that is based on the density c of the copula $\mathscr{C}$ (w.r.t. a reference measure). It is expressed by:

Corollary 3.3.2 (Sklar's theorem - density version)**.** Suppose f is the density of a bivariate (multivariate in a general case) distribution of a pair (X,Y) of random variables and f_1 and f_2 are the marginals of f. Let G_1 and G_2 denote the cumulative marginals of f, then there is a copula function $\mathscr{C}$ whose density c satisfies the following equality:

$$f(x,y) = c(G_1(x), G_2(y)) \times f_1(x) \times f_2(y), \tag{3.8}$$

for any (x,y) in the domain of f.

Reciprocally, if c is the density of a copula and G_1 and G_2 are the univariate cdfs of X and Y, then f is the associated multivariate joint distribution for (X,Y).

This version will be used in our classifier combination model where f will be replaced by one of the K dimensional joint conditional distributions $p(\hat{c}_1,\ldots,\hat{c}_K|y)$. Note that we will use the same copula for any $y \in \Omega$.

Fréchet-Hoeffding Bounds

We present in this subsection two important functions in the context of copula analysis: Fréchet-Hoeffding bounds.

Theorem 2 (Fréchet-Hoeffding bounds)**.** *Let $\mathscr{C}$ denote a 2-dimensional copula, then for every $(x,y) \in [0;1]^2$, $\mathscr{C}$ is bounded by two copulas $\mathscr{M}$ and $\mathscr{W}$, called Fréchet-Hoeffding bounds, resulting in the following inequality:*

$$\mathscr{W}(x,y) = \max(x+y-1,0) \leq \mathscr{C}(x,y) \leq \min(x,y) = \mathscr{M}(x,y). \tag{3.9}$$

Moreover, since $F(x,y) = \mathscr{C}(G_1(x), G_2(y))$ according to theorem 1, we also have that

$$\begin{aligned} &\max(G_1(x) + G_2(y) - 1, 0) \leq F(x,y) \leq \min(G_1(x), G_2(y)), \\ &\Leftrightarrow \mathscr{W}(G_1(x), G_2(y)) \leq F(x,y) \leq \mathscr{M}(G_1(x), G_2(y)) \end{aligned} \tag{3.10}$$

In the K dimensional case ($K > 2$), the function $\mathscr{W}$ is no longer a copula while $\mathscr{M}$ remains so. This theorem is important in the study of copulas because it allows to frame any copula $\mathscr{C}$ (Figure 3.2). Indeed the Fréchet-Hoeffding bounds are featuring specific extreme dependency relations:

- $\mathscr{M}$ is the only possible copula when $X = Y$. In this case, we have

$$p(X = a, Y = b) = \begin{cases} p(X = a) = p(Y = b) & \text{if } a = b \\ 0 & \text{otherwise} \end{cases}. \tag{3.11}$$

  In terms of cumulated distribution, we obtain

$$\begin{aligned} F(a,b) &= p(X \leq a, Y \leq b), && (3.12)\\ &= p(X \leq \min\{a;b\}), && (3.13)\\ &= \min\{p(X \leq a); p(X \leq b)\}, && (3.14)\\ &= \mathscr{M}(a,b). && (3.15) \end{aligned}$$

- Suppose for simplicity that X and Y are binary variables and by convention their codomain is $\{0;1\}$. $\mathscr{W}$ is the only possible copula when $X = 1 - Y$. In this case, we have

$$p(X = a, Y = b) = \begin{cases} p(X = a) = p(Y = b) & \text{if } a \neq b \\ 0 & \text{otherwise} \end{cases}. \tag{3.16}$$

In terms of cumulated distribution, we obtain

$$F(a,b) = p(X \leq a, Y \leq b), \tag{3.17}$$

$$= p(X \leq a, X \geq 1-b), \tag{3.18}$$

$$= \begin{cases} 1 & \text{if } a = b = 1 \\ 0 & \text{if } a = b = 0 \\ p(X=1) & \text{if } a = 1 \text{ and } b = 0, \\ p(X=0) & \text{if } a = 0 \text{ and } b = 1 \end{cases}, \tag{3.19}$$

$$= \mathscr{W}(a,b). \tag{3.20}$$

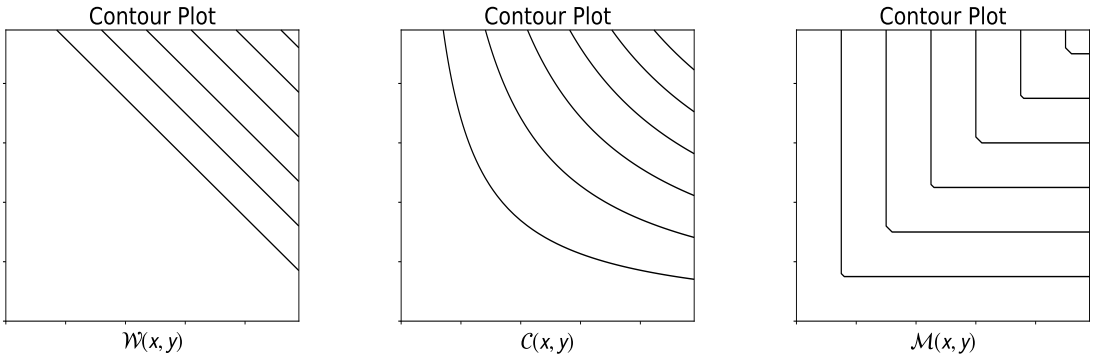


Fig. 3.2 Contour plot of Fréchet-Hoeffding bounds.

As opposed to the extreme dependency relations embodied by the Fréchet-Hoeffding bounds, the independence assumption is captured by the so called independent copula. This copula is such that its density is the constant one function on $[0;1]^2$. Indeed, under this condition, we see that equation (3.8) boils down to the product of the marginals.

3.3.2 Copulas and classifier combination

As outlined in 3.3.1, the popularity of copulas stems from Sklar's theorem (theorem 1). In this theorem, if the cumulative multivariate distribution that we wish to decompose using a copula is continuous, then the copula is unique. However, when we deal with discrete random variables as in our classification problem, the non-uniqueness of the copula raises some identifiability issues [40, 32]. This means that there are many copulas capturing the same statistical relationship between the random variables under study. Without denying

the importance of these issues, we argue that, from a pattern recognition standpoint, what essentially matters is to learn a model that generalizes well. For instance, there are also identifiability issues for neural networks [92] which do not prevent deep nets to achieve state-of-the-art performance in many applications.

In this approach, we investigate parametric copula families to derive a model for the conditional joint distributions $p(\hat{c}_1(\mathbf{X}),\ldots,\hat{c}_K(\mathbf{X})|y)$ where $\mathbf{X}$ is the random vector capturing input uncertainty. Parametric copulas with parameter vector λ are denoted by $\mathscr{C}_\lambda$ but most of the time λ is a scalar and we will work with one such family in the sequel. A difficulty in the quest for an efficient ensemble method is that we must avoid working with cumulative distributions because the computational cost to navigate from cumulative to non-cumulative distributions is prohibitive. Let $\mathbf{G}$ denote a vector whose entries are values of the marginal cdfs of each variable $\hat{c}_i(\mathbf{X})|y$:

$$\mathbf{G}(\mathbf{z}) = \begin{bmatrix} p(\hat{c}_1(\mathbf{X}) \leq z_1|y) \\ \vdots \\ p(\hat{c}_K(\mathbf{X}) \leq z_K|y) \end{bmatrix}, \text{ where } \mathbf{z} = \begin{bmatrix} z_1 \\ \vdots \\ z_K \end{bmatrix} \text{ is a tuple in } \Omega^K. \tag{3.21}$$

We can compute Radon-Nikodym derivatives of $\mathscr{C}_\lambda \circ \mathbf{G}$ w.r.t. a reference measure but again since we work in a discrete setting we will not retrieve closed form expression for the joint distribution we want to infer for an arbitrary large number of classifiers.

As a workaround, we propose to embed each discrete variable $\hat{c}_k(\mathbf{X})|y$ in the real interval $[0;m[$. Let $f_y : \mathbb{R}^K \to \mathbb{R}^+$ be a probability density (w.r.t. Lebesgue) whose support is $[0;m[^K$ and such that for any $\mathbf{z} \in \Omega^K$, we have $f_y(\mathbf{a}) = p(\hat{c}_1(\mathbf{X}) = z_1,\ldots,\hat{c}_K(\mathbf{X}) = z_K|y)$ for any vector $\mathbf{a}$ in the unit volume $\mathscr{V}_\mathbf{z} = [z_1-1;z_1[\times \ldots \times [z_K-1;z_K[$. This means that f_y is piecewise constant and it can be understood as the density of some continuous random vector whose quantized version is equal in distribution to the tuple $(\hat{c}_1(\mathbf{X})|y,\ldots,\hat{c}_K(\mathbf{X})|y)$. Moreover, if $f_y^{(k)}$ is the k^{th} marginal density of f_y, we also have $f_y^{(k)}(a) = p(\hat{c}_k(\mathbf{X}) = z|y)$ for any $a \in [z-1;z[$ and any $z \in \{1;\ldots;m\}$. For any $\mathbf{z} \in \Omega^K$, according to this continuous random vector vision of the problem, we can now thus write

$$p(\hat{c}_1 = z_1,\ldots,\hat{c}_K = z_K|y) = c_\lambda(\mathbf{u}) \times \prod_{k=1}^{K} p(\hat{c}_k = z_k|y), \tag{3.22}$$

$$\mathbf{u} = \left[F_{1,y}(z_1),\ldots,F_{K,y}(z_K)\right] \tag{3.23}$$

where c_λ is the density of $\mathcal{C}_\lambda$ and $F_{k,y}$ is the cumulative distribution of variable $\hat{c}_k(\mathbf{X})|y$. This construction is not dependent on the (arbitrary) way in which the elements of Ω are indexed.

Among parametric copula families, the only one with a closed form density for arbitrary large K is the Gaussian copula. The density of a Gaussian copula [106] is given by

$$\mathcal{C}_\lambda(\mathbf{u}) = \frac{1}{|\mathbf{R}|^{1/2}} \exp\left(-\frac{1}{2}\mathbf{v}^T \cdot \left(\mathbf{R}^{-1} - \mathbf{I}\right) \cdot \mathbf{v}\right), \tag{3.24}$$

where $\mathbf{R}$ is a correlation matrix, $\mathbf{I}$ is the identity matrix and $\mathbf{v}$ is a vector with K entries such that $v_k = Q(u_k)$ where Q is the quantile function of a standard normal distribution. The copula parameters in this case are the correlation matrix entries. Estimating the entries of this matrix is not trivial. We will therefore choose a simplified model and take $\mathbf{R} = \lambda\mathbf{1} + (1-\lambda)\mathbf{I}$ where $\mathbf{1}$ is the all-one matrix. In this model, each diagonal entry of $\mathbf{R}$ is 1 and each non-diagonal entry is λ:

$$\mathbf{R} = \begin{bmatrix} 1 & \lambda & \dots & \lambda \\ \lambda & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \lambda \\ \lambda & \dots & \lambda & 1 \end{bmatrix}. \tag{3.25}$$

The dependency between classifier outputs is regulated by λ which is a scalar living in $\left(\frac{-1}{K-1}; 1\right)$. We also make the assumption that correlation matrices are tied across conditionings on $Y = y$. The $K \times m$ cumulative distributions $F_{k,y}$ are evaluated using estimates of the vectors $[p(\hat{c}_k = c_1|y) \dots p(\hat{c}_k = c_m|y)]^T$ which are drawn from confusion matrices.

Observe that when $\lambda = 0$, the copula density is constant one and the proposed model boils down to the independent case (1.40) which is referred to as the Bayes rule approach in this manuscript.

3.4 Hyperparameter tuning

Now that we have introduced all the ingredients to build our new ensemble method, let us explain how it can be implemented efficiently in practice. The only crucial remaining problem is to tune the parameter λ of the parametric copula. This parameter summarizes the dependency information between each pair of random variables $(\hat{c}_k(\mathbf{X})|y; \hat{c}_{k'}(\mathbf{X})|y)$.

Since we have only one parameter to set, we can use a grid search on the interval $\left(\frac{-1}{m-1}; 1\right)$ using the validation set and select $\hat{\lambda}$ as the value achieving maximal accuracy on this validation set. In the experiments, we use an evenly spaced grid (denoted grid_λ) containing 101 values. The copula-based combination algorithm is given in Algorithm 4. A less formal version of the same procedure is also given in Algorithm 5.

In Algorithm 4, $\mathbb{I}_x$ denotes the indicator function of the singleton $\{x\}$. We also denote by $\boldsymbol{\theta}_y^{(k)}$ the parameter vector of size m of conditional distributions: $\boldsymbol{\theta}_y^{(k)} = \left[\theta_{y,1}^{(k)} \dots \theta_{y,m}^{(k)}\right]^T$ where $\theta_{y,i}^{(k)} = p\left(\hat{c}_k = c_i | y\right)$ and $\boldsymbol{\pi}$ the parameter vector of class distributions of size m: $\boldsymbol{\pi} = [\pi_{c_1}, \dots, \pi_{c_m}]$ where $\pi_{c_i} = p(Y = c_i)$. Using a validation set, The vectors of parameters $\boldsymbol{\pi}$ and $\left\{\boldsymbol{\theta}_1^{(1)}, \dots, \boldsymbol{\theta}_m^{(K)}\right\}$ are estimated using the Laplace add-one smoothing which is the conditional expectation of the parameters given the data in a Dirichlet-multinomial model. As opposed to maximum likelihood estimates, it avoids zero counts which are numerically speaking problematic. It is also recommended to maximize the log-version of (3.1) which is numerically more stable.

Algorithm 4: Copula-based combination model (training)

Data: $\mathscr{D}_{\text{train}}$, n_{val}, grid_λ and $\{\text{train-alg}_k\}_{k=1}^K$

Select n_{val} data points from $\mathscr{D}_{\text{train}}$ to build $\mathscr{D}_{\text{val}}$

$\mathscr{D}'_{\text{train}} \leftarrow \mathscr{D}_{\text{train}} \setminus \mathscr{D}_{\text{val}}$

for $k \in \{1, \dots, K\}$ **do**

  Run train-alg_k on $\mathscr{D}'_{\text{train}}$ to learn $\hat{c}_k$

for $y \in \{c_1, \dots, c_m\}$ **do**

  $\pi_y \leftarrow \frac{1 + \sum_{i=1}^{n_{\text{val}}} \mathbb{I}_y\left(y^{(i)}\right)}{m + n_{\text{val}}}$

  for $k \in \{1, \dots, K\}$ **do**

    for $j \in \{1, \dots, m\}$ **do**

      $\theta_{y,j}^{(k)} \leftarrow \frac{1 + \sum_{i=1}^{n_{\text{val}}} \mathbb{I}_y\left(y^{(i)}\right) \mathbb{I}_{c_j}\left(\hat{c}_k\left(\mathbf{x}^{(i)}\right)\right)}{m + \sum_{i=1}^{n_{\text{val}}} \mathbb{I}_y\left(y^{(i)}\right)}$

      $F_{k,y}\left(c_j\right) \leftarrow \left[1 - \mathbb{I}_{c_1}\left(c_j\right)\right] \times F_{k,y}\left(c_{j-1}\right) + \theta_{y,j}^{(k)}$

for $\lambda \in grid_\lambda$ **do**

  Obtain $\hat{c}$ by pipelining (3.2) and (3.22) using $\hat{c}_1, \dots, \hat{c}_K, \boldsymbol{\pi}, \boldsymbol{\theta}_1^{(1)}, \dots, \boldsymbol{\theta}_m^{(K)}$ and λ

  $\text{Acc}\left(\lambda\right) \leftarrow \frac{\sum_{i=1}^{n_{\text{val}}} \mathbb{I}_{y^{(i)}}\left(\hat{c}\left(\mathbf{x}^{(i)}\right)\right)}{n_{\text{val}}}$

$\hat{\lambda} \leftarrow \underset{\lambda \in \text{grid}_\lambda}{\arg\max}\ \text{Acc}\left(\lambda\right)$

Obtain $\hat{c}$ by pipelining (3.2) and (3.22) using $\hat{c}_1, \dots, \hat{c}_K, \boldsymbol{\pi}, \boldsymbol{\theta}_1^{(1)}, \dots, \boldsymbol{\theta}_m^{(K)}$ and $\hat{\lambda}$

return $\hat{c}$

Finally, one can optionally retrain the classifiers on $\mathscr{D}_{\text{train}}$ after $\hat{\lambda}$ is estimated. Since $\mathscr{D}_{\text{train}}$ is larger then $\mathscr{D}'_{\text{train}} = \mathscr{D}_{\text{train}} \setminus \mathscr{D}_{\text{val}}$, it allows training algorithms to converge to possibly slightly better decision functions. Training them initially on $\mathscr{D}_{\text{train}}$ is however ill-advised as

Algorithm 5: Copula-based combination model (training)

Data: $\mathscr{D}_{\text{train}}$, n_{val}, grid_λ and $\{\text{train-alg}_k\}_{k=1}^{K}$
Select n_{val} data points from $\mathscr{D}_{\text{train}}$ to build $\mathscr{D}_{\text{val}}$
$\mathscr{D}'_{\text{train}} \leftarrow \mathscr{D}_{\text{train}} \setminus \mathscr{D}_{\text{val}}$
for $k \in \{1,\dots,K\}$ **do**
    Run train-alg$_k$ on $\mathscr{D}'_{\text{train}}$ to learn $\hat{c}_k$
for $y \in \{c_1,\dots,c_m\}$ **do**
    Compute prior on probabilities $p(y)$
    for $k \in \{1,\dots,K\}$ **do**
        for $j \in \{1,\dots,m\}$ **do**
            Compute probability distributions $p(\hat{c}_k = c_j|y)$
            Compute cumulative probability distributions $F(\hat{c}_k = c_j|y)$
for $\lambda \in grid_\lambda$ **do**
    Obtain $\hat{c}$ using (3.2) and (3.22) and λ
    Compute accuracy $\text{Acc}(\lambda)$
$\hat{\lambda} \leftarrow \underset{\lambda \in \text{grid}_\lambda}{\arg\max}\ \text{Acc}(\lambda)$
Obtain $\hat{c}$ using (3.2) and (3.22) and $\hat{\lambda}$
return $\hat{c}$

the parameter estimates would be biased. In the next section, where we present numerical results, we use this optional step.

3.5 Application on synthetic and real datasets

In this section, the performance of the copula-based combination approach is assessed in terms of classification accuracy and robustness on both synthetic and real datasets. We also comment on the statistical validation of the results.

3.5.1 Experimental settings

To achieve good assessment of the performance of the proposed combination method, it is necessary to train classifiers in a situation where aggregation has a genuine added value. In a decentralized learning context, desirable experimental conditions to assess combination performances can stated as follows:

(i) There is diversity in the trained base prediction functions $\hat{c}_k$. Indeed, if the base classifiers converge to almost identical functions then the aggregate will not be much different from them either. To ensure a form of diversity, we make the assumption that data is distributed across the network of base classifiers in a non-iid way, that is, each base classifiers only sees inputs that belong to a given region of the feature space. This is a realistic situation as the data stored in a network node might be dependent on the geographic location of this node for instance.

(ii) Base classifiers are *weak*. Since we are evaluating a fusion method, what matters is not to maximize the global classification performance but instead to maximize the performance increment between the base classifiers and the ensemble. One way to allow this is to combine base classifiers with limited capacity, i.e. *weak* classifiers as in boosting [37]. We decided to use logistic regression on each local data set as this algorithm yields a linear decision frontier. Also, logistic regression has the advantage to have no hyperparameter to tune making the conclusions from the experiments immune to this issue. This is also the reason why we do not use a regularized version of this algorithm.

We also need to assess the ability of our approach to be implemented in a decentralized learning setting. In each experiment, we assume that the network load budget is equal to 10% of the overall data. So each node sends 10% of its private data to a central node. The cost of sending functions $\hat{c}_i$ is of several magnitude order lower than the cost of data transfer.

We compare the new approach to the following state-of-the-art or reference methods:

- Bayes rule approach ((1.40)).
- classifier selection based on accuracies,
- maximally accurate individual classifier,
- weighted vote combination based on accuracies,
- meta classifier trained using stacking,
- and a centralized classifier trained on all data.

Each method relying on base classifier accuracies uses the data sent on the central node as validation set to estimate these accuracies. The validation set is also used as part of stacking to generate inputs for the second stage training. We also use a logistic regression

for this second stage and input entries are predicted classes from each base classifier. The maximally accurate individual classifier and the centralized classifier are not applicable in the decentralized setting but they are relevant references as part of a benchmark for comparison. Concerning the copula-based model, we examine the simplified Gaussian copula where the copula hyperparameter is estimated by grid search from the validation set.

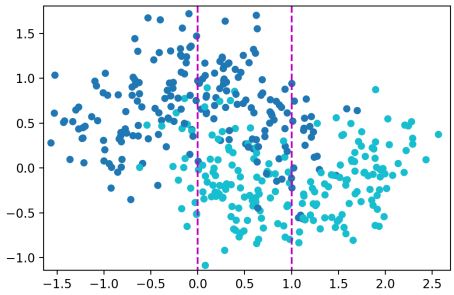
3.5.2 Experiments on synthetic data

Using synthetic data sets is advantageous in the sense that, in the test phase, we can generate as many data as we want to obtain very reliable estimates of classification accuracies. We examine three different data generation processes from `sklearn` python library: Moons, Blobs and Circles. Each of these processes yields non-linearly separable data sets as illustrated in Figure 3.3.

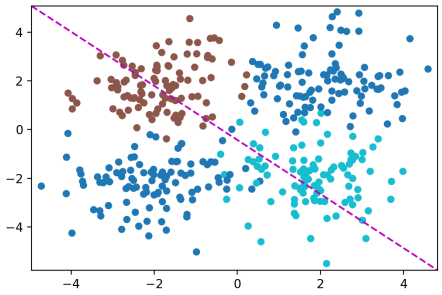
Dataset specifications and generation procedure

The Moons and Circles data sets are binary classification problems while Blobs involves three classes. For each problem, the data set is partitioned into disjoint regions of the input space as specified in Figure 3.3 and consequently we combine two base classifiers for the Blobs data set and three base classifiers for the others. Also, in each case, input vectors live in $\mathbb{R}^2$.

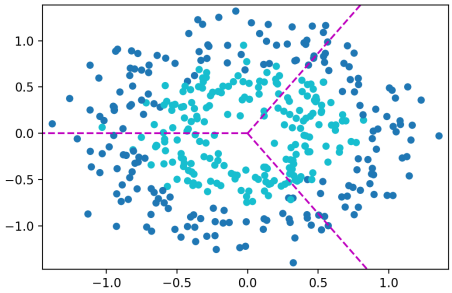
The Moons data set consists in two half-circles to which a Gaussian noise is added. For each half-circle, one of its extremal point is the center of the other half-circle. The covariance matrix of the noise in our experiment is $0.3 \times \mathbf{I}$ where $\mathbf{I}$ is the identity matrix. Before adding this noise, we also randomized the position of sample points on the half circle using a uniform distribution while the baseline `sklearn` function samples such points with fixed angle step. The Blobs data set is also obtained using a slightly different function than its `sklearn` version. It generates a data set from four 2D Gaussian distributions centered on each corner of a centered square whose edge length is 4. Each distribution covariance matrix is $\mathbf{I}$. The examples generated by the distributions whose expectations are $(-2;-2)$ and $(2;2)$ are assigned to class c_1. Each remaining Gaussian distribution yields examples for either class c_2 or c_3. Finally, the Circles data set consists in sampling with fixed angle step two series of points from centered circles with radius 0.5 and 1. A Gaussian noise with covariance matrix $0.15 \times \mathbf{I}$ is added to these points.



(a) Moons



(b) Blobs



(c) Circles

Fig. 3.3 Synthetic data sets and their partitions into feature space regions.

Combination method performances

To evaluate the accuracy of a classifier or classifier ensemble trained on a data set drawn from any of the above mentioned generating processes, we drew test points from the same process until the Clopper-Pearson confidence interval of the accuracy has length below 0.2% with confidence probability 0.95. For each generating process, we repeated this procedure 3000 times to estimate the expected accuracy across data set draws.

The estimated expected accuracies and the estimated accuracy standard deviations are given for each classification method of the benchmark in Tables 3.1 and 3.2 for $n_{\text{train}} = 200$ and $n_{\text{train}} = 400$ respectively. Best performances are in bold except for the accuracies of the optimal classifier (which is always the highest) that are provided as indicators of the best performance that can be achieved on these datasets. In these experiments, the copula-based method is the top 1 for the Blobs and Circles data sets. For the Moons dataset, it is the top 2 method when $n = 400$ and top 3 when $n = 200$. Observe that when $n = 200$, the top 2 method is the Bayes rule approach. Since the copula based approach generalizes this latter, we can assume that the validation set was not big enough to select a value of λ that at least

Method	**Moons**	**Blobs**	**Circles**
Gauss. Copula	80.57 ± 4.68	$\mathbf{93.15} \pm 4.83$	$\mathbf{84.49} \pm 4.51$
Bayes rule	83.46 ± 2.91	91.14 ± 7.27	79.32 ± 6.70
Stacking	81.07 ± 3.89	69.87 ± 5.37	70.20 ± 8.08
Weighted Vote	84.60 ± 2.20	82.43 ± 11.02	50.50 ± 0.05
Clf. Selection	79.25 ± 3.51	72.34 ± 0.37	62.38 ± 0.32
Max. Ind. Clf.	79.25 ± 1.67	72.34 ± 0.36	62.38 ± 0.32
Centralized Clf.	$\mathbf{84.99} \pm 0.55$	88.49 ± 0.42	50.02 ± 0.49
Optimal Clf.	91.50 ± 0.0	95.50 ± 0.0	94.50 ± 0.0

Table 3.1 Classification accuracies for several synthetic data sets and several classifier or classifier ensembles. ($n_{\text{train}} = 200$)

Method	**Moons**	**Blobs**	**Circles**
Gauss. Copula	86.75 ± 3.07	$\mathbf{94.39} \pm 0.96$	$\mathbf{86.39} \pm 1.11$
Bayes rule	86.43 ± 3.28	93.78 ± 2.48	84.54 ± 4.45
Stacking	85.32 ± 4.08	71.70 ± 2.61	78.19 ± 6.95
Weighted Vote	$\mathbf{87.83} \pm 1.19$	78.72 ± 9.96	50.50 ± 0.0
Clf. Selection	79.67 ± 2.14	72.43 ± 0.27	62.50 ± 0.05
Max. Ind. Clf.	80.66 ± 1.08	72.45 ± 0.22	62.50 ± 0.06
Centralized Clf.	85.22 ± 0.45	88.72 ± 0.42	50.01 ± 0.5
Optimal Clf.	91.50 ± 0.0	95.50 ± 0.0	94.50 ± 0.0

Table 3.2 Classification accuracies for several synthetic data sets and several classifier or classifier ensembles. ($n_{\text{train}} = 400$)

makes the copula based approach performing as well as Bayes rule. Most importantly, the copula based method and the Bayes rule approach are obviously more robust since they never perform poorly on any data set. While the weighted vote method is the top 1 for the Moons data set, it completely crashed on the Circles data set and converges to a random classifier.

Another result which is surprising at first sight, is that the centralized classifier is sometimes outperformed by some decentralized ensembles. This is actually well explained by the deterministic way in which input spaces are partitioned. Indeed, the partitions are cleverly

chosen so that a combination of linear decision frontiers fits intuitively a lot better the data than a single linear separation does. In other words, ensembles have a larger VC dimension[2] and visit a larger hypotheses set. One may wonder to which extent it would be possible to purposely partition data sets in such a relevant way to reproduce such conditions in more general situations. This is however beyond the scope of the experiments presented in this chapter in which we address decentralized learning, a setting where we take distributed data as is and we cannot reorganize them.

3.5.3 Experiments on real data

To upraise the ability of the benchmarked methods to be deployed in a decentralized learning setting, we also need to test them on sets of real data. This subsection presents a number of such experiments.

Dataset specifications

Since decentralized learning is essentially useful in a big data context, we chose three rather large public data sets: 20newsgroup, MNIST and Satellite. The specifications of these data sets are reported in Table 3.3. Example entries from the 20newsgroup data set are word counts obtained using the term frequency - inverse document frequency statistics. We reduced the dimensionality of inputs using a latent semantic analysis [20] which is a standard practice for text data. We kept 100 dimensions. Also, as recommended, we stripped out each text from headers, footers and quotes which lead to overfitting. For both MNIST and Satellite, we kept them unchanged.

Datasets	# samples	# features	# classes	Input type	Class type	Source
20newsgroup	18846	100 after red.	20	text	text topic	`Sklearn`
		101631 before red.				
MNIST	70000	784	10	image	digit	`Sklearn`
Satellite	6435	36	6	multi-spectral image features	soil type	UCI repository

Table 3.3 Real data set specifications

[2]The Vapnik-Chervonenkis (VC) dimension is an integer featuring the ability of decision functions learned by a training algorithm to shatter a finite number of data points. When the VC dimension is high, an algorithm can learn complex decision functions.

Implemented experimental protocol

Unlike synthetic data sets, we need to separate the original data set into a train set and a test set. To avoid a dependency of the reported performance w.r.t train/test splits, we perform 2-fold cross validation (CV). Also, we shuffle at random examples and repeat the training and test phases 500 times.

To comply with the diversity condition, we distributed the training data over network nodes using the following procedure: for each data set, for each class,

1. apply principal component analysis to the corresponding data,
2. project this data on the dimension with highest eigenvalue,
3. sort the projected values and split them into K subsets of cardinality n_i/K where n_i is the proportion of examples belonging to class c_i.

For each subset of the data assigned to the same class, each base classifier has access to examples in a region that is disjoint from those accessed by the other classifiers. We argue that this way of splitting data is somehow adversarial because some nodes may see data that are a lot easier to separate than it should and will consequently not generalize very well.

Combination method performances

Average accuracies over random shuffles and CV-folds are given in Tables 3.4, 3.5 and 3.6 for $K = 2$, 10 and 50 nodes respectively. Best accuracies are in bold font in the tables. Note that for real datasets the optimal classifier does not exist and thus it cannot be found in the tables of results.

In these experiments, decentralized ensemble methods have difficulties to compete with a centralized classifier except for the copula-based method or the Bayes rule approach when K is rather large. This is presumably because PCA-based data splits do not allow to discover better decision frontiers. We observe that the weighted vote ensemble, the Bayes rule approach and the copula-based ensemble have a tendency to achieve higher accuracies as K increases. An exception to this conclusion is the Satellite data with $K = 50$ nodes. Remember that given that we use a 2-fold-CV, each node has access to 64 data points only in this case while they must learn 101 parameters. So it is not surprising that, after some point, increasing K is at the expense of the ability of base classifiers to avoid overfitting. As opposed to ensemble methods, classifier selection seems to be more efficient when K is small which is not adapted to a decentralized learning setting.

Method	**20newsgroup**	**MNIST**	**Satellite**
Gauss. Copula	47.96 ± 1.2	68.68 ± 2.10	78.29 ± 2.19
Bayes rule	48.08 ± 1.16	68.42 ± 2.04	78.66 ± 2.13
Stacking	17.09 ± 3.38	37.83 ± 4.12	64.02 ± 2.18
Weighted Vote	47.97 ± 1.25	66.71 ± 1.70	78.05 ± 1.81
Clf. Selection	47.96 ± 1.25	66.71 ± 1.70	78.05 ± 1.81
Max. Ind. Clf.	48.87 ± 0.88	67.23 ± 1.46	79.25 ± 1.38
Centralized Clf.	$\mathbf{58.19} \pm 0.36$	$\mathbf{90.65} \pm 0.33$	$\mathbf{83.16} \pm 0.40$

Table 3.4 Classification accuracies for several real data sets and several classifier or classifier ensembles. ($K = 2$ classifiers)

Method	**20newsgroup**	**MNIST**	**Satellite**
Gauss. Copula	49.06 ± 0.64	85.86 ± 1.17	82.99 ± 0.83
Bayes rule	49.19 ± 0.64	85.77 ± 1.30	$\mathbf{83.21} \pm 0.68$
Stacking	14.47 ± 1.13	41.47 ± 2.90	70.16 ± 3.35
Weighted Vote	50.17 ± 0.65	82.46 ± 1.54	81.99 ± 0.80
Clf. Selection	37.35 ± 1.38	66.26 ± 1.57	77.83 ± 2.04
Max. Ind. Clf.	38.25 ± 0.68	67.24 ± 0.76	79.10 ± 1.16
Centralized Clf.	$\mathbf{58.19} \pm 0.36$	$\mathbf{90.65} \pm 0.33$	83.16 ± 0.40

Table 3.5 Classification accuracies for several real data sets and several classifier or classifier ensembles. ($K = 10$ classifiers)

Most importantly, we see that the copula-based method does not perform poorly on any dataset as compared to other decentralized approaches which is in line with the robustness observed in the synthetic data set experiments.

3.5.4 Statistical validation

To draw reliable conclusions based on the experiments on synthetic and real data in the previous subsection, we apply a Wilcoxon signed-rank test as in chapter 2. We validate the significance of performance discrepancies between the Gaussian copula-based fusion method and concurrent approaches (Bayes rule, stacking, weighted vote, selected classifier,

Method	**20newsgroup**	**MNIST**	**Satellite**
Gauss. Copula	50.16 ± 0.69	87.83 ± 0.93	75.04 ± 1.06
Bayes rule	50.26 ± 0.61	87.78 ± 0.96	75.08 ± 0.96
Stacking	17.95 ± 0.92	55.56 ± 1.84	55.23 ± 2.76
Weighted Vote	52.06 ± 0.46	84.45 ± 1.34	75.61 ± 0.62
Clf. Selection	34.89 ± 1.25	69.30 ± 1.40	63.73 ± 2.66
Max. Ind. Clf.	35.90 ± 0.65	69.64 ± 1.23	66.33 ± 1.43
Centralized Clf.	$\mathbf{58.19} \pm 0.36$	$\mathbf{90.65} \pm 0.33$	$\mathbf{83.16} \pm 0.40$

Table 3.6 Classification accuracies for several real data sets and several classifier or classifier ensembles. ($K = 50$ classifiers)

maximally accurate individual classier, centralized classifier and optimal[3]). The detailed tables are presented in Appendix D. We kept the same form of the tables as those of chapter 2. We also kept the threshold of 5% for the p-value to reject the null hypothesis.

Pairs of methods	# ✓	# ✗	#t-norm wins	#t-norm loses	#tie
Gauss. copula *vs.* Bayes rule	15	0	8	7	0
Gauss. copula *vs.* Stacking	15	0	14	1	0
Gauss. copula *vs.* Weighted vote	14	0	12	2	0
Gauss. copula *vs.* Clf. Selection	14	1	14	0	0
Gauss. copula *vs.* Max. Ind. Clf.	15	0	12	3	0
Gauss. copula *vs.* Centralized Clf.	15	0	5	10	0
Gauss. copula *vs.* optimal Clf.	6	0	0	6	0

Table 3.7 Synthetic summary of Wilcoxon tests. Detailed tables are found in D.

Table 3.7 summarizes pairwise comparisons between the Gaussian copula-based probabilistic approach to each concurrent one. Except for the optimal classifier which is unreachable in reality, we notice that the proposed approach achieves statistically significant higher accuracies and thus the conclusions of the previous subsection are confirmed.

[3]For only synthetic datasets.

3.5.5 Comments on the copula type

In both synthetic and real data sets, the Bayes rule approach and the Gaussian copula-based method achieve comparable accuracies most of the time which seems to suggests that using the Gaussian copula one has a limited interest. There are three situations in which significant performance discrepancies are observed. The first one is the Moons data set when $n_{\text{train}} = 200$. We argue that the Gaussian copula-based ensemble fails to correctly estimate its parameter λ as performance levels are reversed when $n_{\text{train}} = 400$ and the validation set has now 40 elements instead of 20.

The other situations are the Circles data set when either $n_{\text{train}} = 200$ or $n_{\text{train}} = 400$. In this case, we see that the Bayes rule approach fails to keep up with the Gaussian copula one regardless of how many points the validation set contains. In conclusion, the Gaussian model does offer increased robustness as compared to the independent one provided that the validation set size allows to tune correctly λ. Remember that when $\lambda = 0$, both models coincide, so if we have enough data and if being independent is really what works best, then there is no reason why we should not obtain $\hat{\lambda} = 0$.

3.5.6 Comparison between the possibilistic and the probabilistic approaches

In this section, we proceed to a comparison between the two parametrized approaches proposed in thesis (chapter 2 and 3). The first approach is a t-norm based combination method that lies in the possibilistic framework while the second one is a Gaussian copula-based combination method that lies in the probabilistic framework. For both approaches , the inputs are contextual data extracted from confusion matrices and both of them are controlled by a parameter that allows the adaptation of the combination rule to classifier dependencies. We present the comparison in two subsections. In the first one, we compare our two contributions on the basis of the experiments carried out in chapter 2 and in the second one we compare them on the basis of the experiments carried out in this very chapter. A statistical validation is provided in Appendix E.

Comparison with respect to chapter 2

In the following paragraphs, we use the same experimental protocol as in chapter 2 where three classifiers are trained on mutually exclusive sets of attributes. Details of the experimental protocol are found in section 2.4.4. We tested the copula-based approach on the same

datasets used in the experiments of the previous chapter (Digits, Waveform, Wine, Cancer, CNAE, Segments and MNIST) using three classifier ensembles of different base classifiers (Classification trees, logistic regressions and Gaussian naive Bayes classifiers).

Results of classification trees, logistic regressions and naive Bayes classifiers are presented in tables 3.8, 3.9 and 3.10 respectively.

Method	**Digits**	**Waveform**	**Cancer**	**CNAE**	**Segments**	**Wine**	**MNIST**
Poss. t-norm	**67.85** $\pm$ 3.38	**75.23** $\pm$ 1.08	**90.76** $\pm$ 2.82	47.07 $\pm$ 4.38	**73.63** $\pm$ 5.75	**88.79** $\pm$ 4.04	**62.62** $\pm$ 1.65
Gauss. copula	65.86 $\pm$ 4.68	62.16 $\pm$ 5.17	89.01 $\pm$ 3.93	**54.05** $\pm$ 4.95	72.17 $\pm$ 6.21	80.31 $\pm$ 4.06	60.39 $\pm$ 0.016

Table 3.8 Average accuracies ($\pm$ standard deviations) of the combination of three classification trees over 100 iterations.

Method	**Digits**	**Waveform**	**Cancer**	**CNAE**	**Segments**	**Wine**	**MNIST**
Poss. t-norm	**89.91** $\pm$ 1.27	**83.24** $\pm$ 0.64	**99.81** $\pm$ 0.06	**84.08** $\pm$ 2.21	**90.70** $\pm$ 1.23	**88.85** $\pm$ 3.40	**88.20** $\pm$ 0.0700
Gauss. copula	79.94 $\pm$ 1.37	69.50 $\pm$ 4.95	90.39 $\pm$ 1.17	77.14 $\pm$ 2.73	80.51 $\pm$ 4.94	79.47 $\pm$ 5.88	77.87 $\pm$ 0.0004

Table 3.9 Average accuracies ($\pm$ standard deviations) of the combination of three logistic regression classifiers over 100 iterations.

Method	**Digits**	**Waveform**	**Cancer**	**CNAE**	**Segments**	**Wine**	**MNIST**
Poss. t-norm	**79.13** $\pm$ 2.47	**79.59** $\pm$ 0.81	**83.46** $\pm$ 2.00	**78.93** $\pm$ 2.81	**84.21** $\pm$ 2.13	**92.10** $\pm$ 2.71	**64.64** $\pm$1.110
Gauss. copula	78.55 $\pm$ 2.50	71.12 $\pm$ 5.41	82.52 $\pm$ 2.05	76.63 $\pm$ 2.11	78.73 $\pm$ 5.52	84.76 $\pm$ 5.23	56.21 $\pm$ 0.152

Table 3.10 Average accuracies ($\pm$ standard deviations) of the combination of three naive Bayesian classifiers with Gaussian class conditional distribution over 100 iterations.

Comparison with respect to chapter 3

In the following paragraphs, we use the same experimental protocol as in chapter 3 where different logistic regression ensembles are combined in a decentralized setting. We also keep both real and synthetic datasets described in 3.5.3 and 3.5.2.

Results of logistic regression ensembles for synthetic and real datasets are shown in tables 3.11, 3.12, 3.13, 3.14 and 3.15 respectively.

Method	**Moons**	**Blobs**	**Circles**
Poss. t-norm	**81.59** $\pm$ 0.03	87.44 $\pm$ 0.106	81.62 $\pm$ 0.065
Gauss. Copula	80.57 $\pm$ 4.68	**93.15** $\pm$ 4.830	**84.49** $\pm$ 4.510

Table 3.11 Classification accuracies for several synthetic data sets and several classifier or classifier ensembles. ($n_{\text{train}} = 200$)

Method	**Moons**	**Blobs**	**Circles**
Poss. t-norm	86.68 $\pm$ 0.03	93.70 $\pm$ 0.04	85.68 $\pm$ 0.03
Gauss. Copula	**86.75** $\pm$ 3.07	**94.39** $\pm$ 0.96	**86.39** $\pm$ 1.11

Table 3.12 Classification accuracies for several synthetic data sets and several classifier or classifier ensembles. ($n_{\text{train}} = 400$)

Method	**20newsgroup**	**MNIST**	**Satellite**
Poss. t-norm	45.91 $\pm$ 0.01	67.93 $\pm$ 0.02	77.76 $\pm$ 0.02
Gauss. Copula	**47.96** $\pm$ 1.20	**68.68** $\pm$ 2.10	**78.29** $\pm$ 2.19

Table 3.13 Classification accuracies for several real data sets and several classifier or classifier ensembles. ($K = 2$ classifiers)

Method	**20newsgroup**	**MNIST**	**Satellite**
Poss. t-norm	38.87 $\pm$ 0.02	84.96 $\pm$ 0.013	79.55 $\pm$ 0.014
Gauss. Copula	**49.06** $\pm$ 0.64	**85.86** $\pm$ 1.17	**82.99** $\pm$ 0.83

Table 3.14 Classification accuracies for several real data sets and several classifier or classifier ensembles. ($K = 10$ classifiers)

Method	**20newsgroup**	**MNIST**	**Satellite**
Poss. t-norm	35.66 $\pm$ 0.022	86.90 $\pm$ 0.01	71.03 $\pm$ 0.036
Gauss. Copula	**50.16** $\pm$ 0.69	**87.83** $\pm$ 0.93	**75.04** $\pm$ 1.06

Table 3.15 Classification accuracies for several real data sets and several classifier or classifier ensembles. ($K = 50$ classifiers)

Comments on the comparison results

In the previous subsections, we performed comparisons between the two proposed approaches introduced as part of this seminal works (the t-norm possibilistic approach and the Gaussian copula-based approach). According to result tables, we remark that the possibilistic approach

outperforms the copula based combination when the dataset is divided with respect to feature indices while the copula based combination seems to achieve higher accuracies when the dataset is divided with respect to example indices.

An important conclusion that can be drawn from this is that none of the proposed approaches subsumes the other one, i.e. we know that there are circumstances in which one of them is more relevant than the other and conversely. Unfortunately, we are not able to precisely (or at least roughly) identify those situations in which the copula based method should be preferred as well as those situations in which the t-norm based method should be preferred. Intuitively, we cannot generalize the conclusion of the above comparison in terms of feature/example splits of the dataset. Indeed, remember that in the experiments where the dataset is split the Bayes rule approach outperforms the t-norm based method and that the copula based method encompasses the Bayes rule approach. The moderate performances of the copula based approach in these experiments is thus probably explained by an inaccurate estimation of the copula parameter λ.

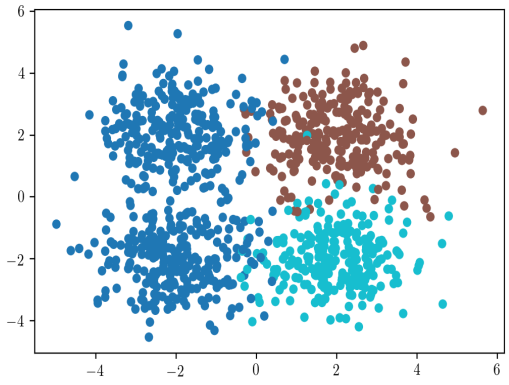
However, we can comment on the sensitivity of both methods with respect to their main hyperparameter λ. The monotonicity of the function that carries out the fusion (either the t-norm or the copula) with respect to this parameter plays a major role in the classification performances because a minor modification of the parameter value (such as a bad estimation of the parameter for instance) may have a strong impact on the final decision of the combination method. The following example is an attempt to get insights regarding this question.

Example 5. In this example, we examine the dataset displayed in Figure 3.4a. This dataset is similar to the Blobs dataset that we used in the experiments on synthetic data that are reported in this chapter. This dataset is also sampled from four 2D Gaussian distributions, each of which is centered on a corner of a centered square whose edge length is equal to 4. The same number of examples is generated by each Gaussian distribution. However, in the dataset examined in this example the input space region corresponding to the blue class is easier to characterize. Indeed, one only needs to examine the 1st entry of an example and if this latter is negative then we must assign it to the blue class.

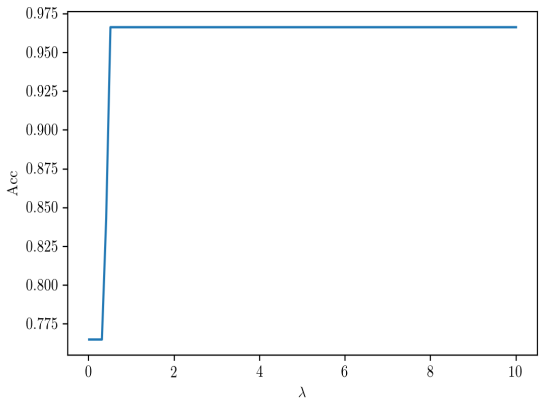
Suppose that two classifiers $\hat{c}_1$ and $\hat{c}_2$ were trained respectively using only the first or only the second entry of training examples. Suppose also that they converged to the following decision functions:

- $\hat{c}_1$ assigns an example to the blue class whenever $x_1 < 0$. Otherwise, it can randomly assigns a test sample to either the brown or the cyan class. So the accuracy of $\hat{c}_1$ is around 0.75.
- $\hat{c}_2$ assigns an example to the cyan class whenever $x_2 < 0$. Otherwise, it assigns a test sample to the brown class and consequently it always fails to recognize members of the blue class. Its accuracy is thus around 0.5.

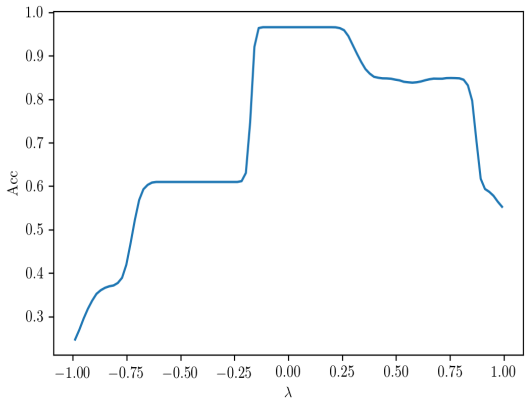
Figures 3.4b and 3.4b give the accuracies as functions of λ obtained by combining $\hat{c}_1$ and $\hat{c}_2$ using the t-norm possibilistic approach and the copula based approach respectively.



(a) A simple 2D dataset with three classes. The optimal classification rule consists in assigning the blue class to examples such that $x_1 < 0$, the brown class when x_1 and $x_2 > 0$ and the cyan class otherwise.



(b) Parametrized Aczel Alsina t-norm accuracy w.rt. λ.



(c) Parametrized Gaussian copula accuracy w.rt. λ.

In Figure 3.4c, we notice that the best accuracy is achieved when λ is close to 0, which is what we expected because the Gaussian distributions have diagonal covariance matrices and thus $\hat{c}_1(X)$ and $\hat{c}_2(X)$ are independent random variables. In Figure 3.4b, we observe that the multiplicative rule ($\lambda = 1$) is one of the value that achieves the best accuracy. Obviously, the range of values of λ that achieves maximal accuracy is larger for the t-norm than for the copula.

Two different standpoints can be advocated. On the one hand, we can conclude that the t-norm based approach is more robust to incorrect adjustments of the hyperparameter

λ in the sense that moderate variations of this latter will not translate into poorly accurate aggregated decision functions. On the other hand, we see that the set of learnable functions for the t-norm based approach is more limited than for the copula based approach. So the copula based approach has apparently a greater capacity. Altough one example is not enough to completely solve this question, it seems that when the dataset is not very large, the t-norm based approach should be preferred over the copula based approach because in this situation it is likely that a cross validation grid search will not produce a very reliable estimate of λ. Conversely, when the dataset is large enough, the copula based approach should be preferred because it allows to examine a larger number of aggregated decision functions. Finally, this comparison confirms the robustness of the t-norm possibilistic approach to parameter estimation which is in line with the previous conclusions in chapter 2.

3.6 Conclusion

This chapter presents the second original contribution of this our seminal works. In this chapter, we introduce a new ensemble method that relies on a probabilistic model. Given a set of trained classifiers, we evaluate the probabilities of each classifier output given the true class on a validation set. We use a Gaussian copula to retrieve the joint conditional distributions of these latter which allow us to build an ensemble decision function that consists in maximizing the probability of the true class given all classifier outputs.

We assess the benefits this new approach by showing that it fits a decentralized learning setting which is a modern concern in a big data context. The approach is validated through numerical experiments on both synthetic and real data sets. We show that a Gaussian copula based ensemble achieves higher robustness than other ensemble techniques and can compete or outperform a centralized learning in some situations. A statistical validation study was carried out using Wilcoxon sign-ranked test. The results report a statistically significant accuracy increment for our Gaussian copula-based approach to dependent classifier combination as compared to 3 other combination methods and to classifier selection. The tuning of the parameter of the copula can be done using cross validation without user supervision.

Conclusions and Perspectives

Outcomes of the thesis

In these seminal works, we addressed the problem of constructing robust combination methods for dependent classifiers that are suitable in a decentralized setting, i.e. when the data is spread across nodes in a network and we can only exchange a limited fraction of the subsets stored in a each node. Each node may have access to a limited number of features or to a subset of the training examples or both. The robustness that is sought consists in achieving high accuracy levels regardless of the way the training data is distributed in the network.

To achieve this task, we set two specifications: the first one consists in using contextual information (extracted from confusion matrices) that reflects the confidence level that can be given to an individual classifier within a classifier ensemble while the second one deals with the ability to adapt the combination rule to the level of statistical dependency in classifier decisions. The problem was studied within two different uncertainty frameworks in which we proposed two original combination schemes (chapters 2 and 3) that are compliant with the previous specifications. These two approaches are the main contributions of this thesis. For both approaches, the starting point of our reasoning stems from Bayes rule that allows the computation of the joint conditional distribution of classifier predictions given the true label. A simple way to compute this distribution relies on condtional independence assumptions allowing to decompose it into a product of the marginal conditional distributions and the class distribution. This procedure is referred to as the Bayes rule approach in this manuscript. Although these hypotheses make the computation easier and takes into account individual classifier performances, they are patently unrealistic. Indeed, most classifiers (when trained independently) will produce the same predictions in the same regions of the input space and are consequently obviouly dependent.

When the classifiers are independent, a product-based combination rule such as the Bayes rule approach is very suitable since it considers that the information given by the classifiers

are complementary and if many classifiers are confident that example $\mathbf{x}$ should be classified as a member of class y, then our confidence after combination should be higher than those of the individual classifiers. We call a combination of this kind, a rule that allows reinforcement. However, if the classifiers are highly dependent a recommended combination rule is an idempotent one, meaning that it does not capture the same information twice so if several classifiers have the same level of confidence that example $\mathbf{x}$ should be classified as a member of class y, then after combination our level of confidence is no higher. Thus, to correctly model the dependence between classifiers we proposed two approaches that are based on parametrized combination rules which allow more flexibility in the models by visiting a continuum of rules ranging from indempotent ones to reinforcing ones.

The manuscript was organized in three chapters. In chapter 1, we presented a short state of the art of classification algorithms. We then presented the motivations behind classifier combination. Those motivations are related to some limitations when using a single classifier which are mainly dealing with statistical, computational and representational aspects. We also reviewed thoroughly classifier combination approaches which are sorted in the following categories: classical combination algorithms, ensemble learning and trainable fusion methods. Finally, we exposed a state of the art of classifier combination within different uncertainty frameworks such as probability, belief functions and fuzzy set theories. In the second and third chapters, we presented the proposed combination methods. In both chapters, we first explained the theoretical foundations of the combination approach and then we carried out some experiments to validate the efficiency of the new method. We note that both approaches are not restricted to given type of classifiers (heterogeneous combination).

The first main contribution is a novel classifier combination approach designed in the possibilistic framework. The new method is based on a parametrized t-norm rule, namely the Aczel-Alsina t-norm family, that combines contextual information represented by conditional probabilities of the true class label given the label prediction issued by a base classifier. The experiments in chapter 2, demonstrated the robustness of the new method as a dependent classifiers combination model and, especially as a fault tolerant approach (when irrelevant classifiers are added in the ensemble). We believe that thanks to the t-norm parameter the dependency between classifiers is modeled efficiently. In this possibilistic approach, the t-norm parameter allows to navigate between the product rule and the minimum rules which are suitable in the cases of low and high dependence respectively. The parameter is tuned by a cross validation on a grid search.

The second main contribution is another novel classifier combination approach designed in the probabilistic framework. It is based on a parametrized Gaussian copula to model statis-

tical dependency between classifier decisions. In this approach, we build the combination model by relying on Sklar's theorem that demands two types of contextual information: the estimated conditional probabilities of the predictions given the true label and a multivariate function having necessary properties to qualify as a copula. Thanks to this theorem, we are able to estimate the joint conditional probability that is the core of the decision rule. We carried out several experiments involving the new method and it showed its efficiency as compared to benchmarked approaches. The parameter of the Gaussian copula was tuned by cross validation in a similar way as for the possibilistic approach. The reported performances induced by this approach have shown increased sensitivity with respect to the parameter of the rule than the t-norm approach.

By the end of this manuscript, we attain the main objective of this thesis that consists in developing robust combination approaches that are able to apparaise statistical dependency between classifiers and are applicable in a decentralized setting. However, there is room for improvements and a number of aspects of the proposed approaches deserves to be examined in future research works.

Perspectives

At the end of these seminal works, several research directions can be envisaged. Some directions are immediate extensions of the proposed approaches while others are more general and concern other combination schemes as well.

Perspectives related to the proposed approaches

As any model, the proposed approaches have a number of limitations and can be improved with respect to several aspects. We believe that future works should be focused on the following points:

- As previously indicated, the t-norm and the copula based approaches depends on a parameter λ that is able to capture the statistical dependency between classifiers. However, in our experiments we used a single parameter to combine all classifiers in the ensemble which is not optimal since there may exist different levels of dependency between different subsets of classifiers. For instance, given a panel of 5 classifiers in which 3 are highly dependent and the others are independent, a single parameter will surely find it difficult to capture the two levels of dependency. To solve this issue, one can target a sequential or a hierarchical combination procedure. In a sequential combination of classifiers, two classifiers are first combined, then the resulting classifier is combined with the third classifier in the ensemble and so on. In a hierarchical combination process, we first separate the classifiers in different sets according to their level of dependency then we combine the classifiers of each set and finally we re-combine the resulting classifiers. This idea is somewhat similar to an approach by Quost et al. [81] in which they used unsupervised techniques to cluster classifiers thereby deriving the desired hierarchy. A challenge in both these extensions of our contributions is that several parameters need to be estimated and a mere grid search does no longer suffice.

- The approaches presented in this thesis are global methods meaning that they use the same combination rule for all test samples which is advantageous in terms of the simplicity as compared to local methods where the contributions of each classifier individually is input dependent. However, in order to obtain better performances, local information is intuitively beneficial. There are several possibilities to integrate local information in the combination process. Generally speaking, we would like to obtain probabilities $p\left(y|\hat{c}_1\left(\mathbf{x}\right),\ldots,\hat{c}_K\left(\mathbf{x}\right),\mathbf{x}\right)$ instead of $p\left(y|\hat{c}_1\left(\mathbf{x}\right),\ldots,\hat{c}_K\left(\mathbf{x}\right)\right)$ as in

our approaches. If the classifiers are probabilistic, their output gives access to estimates of $p(y|\mathbf{x})$. By restricting the type of classifiers in the ensemble to probabilistic ones, it seems feasible to use local information.

General perspectives related to classifier combination

The last perspective in the wake of these seminal works deals with classifier combination and deep learning. Indeed, deep neural networks have proved in many applicative fields that they can efficiently learn complex decision functions (with many non linearities). An interesting research direction is to use deep nets as the second stage classifier in a stacking approach. This second stage classifier can be fed with inputs whose entries would be classifier decisions, contextual information or other types of classifier-related data. The difficulty in this regard is to derive an efficient network architecture. Intuively, the architecture should process in a different way categorical data such as classifier decisions and continuous data such as classifier confidence levels.

References

[yif] MS Windows NT kernel description. http://blog.pengyifan.com/tikz-example-svm-trained-with-samples-from-two-classes. Accessed: 2010-09-30.

[Mar] MS Windows NT kernel description. https://tex.stackexchange.com/questions/153957/drawing-neural-network-with-tikz. Accessed: 2010-09-30.

[3] Al-Ani, A. and Deriche, M. (2002). A new technique for combining multiple classifiers using the dempster-shafer theory of evidence. *Journal of Artificial Intelligence Research*, 17:333–361.

[4] Allard, D., Comunian, A., and Renard, P. (2012). Probability aggregation methods in geoscience. *Mathematical Geosciences*, 44(5):545–581.

[5] Baruque, B., Porras, S., and Corchado, E. (2011). Hybrid classification ensemble using topology-preserving clustering. *New Generation Computing*, 29(3):329.

[6] Bernard, S. (2009). *Forêts Aléatoires: De l'Analyse des Mécanismes de Fonctionnement à la Construction Dynamique*. PhD thesis, Université de Rouen.

[7] Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2):123–140.

[8] Breiman, L. (1999). Using adaptive bagging to debias regressions. Technical report, Technical Report 547, Statistics Dept. UCB.

[9] Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

[10] Brown, G., Wyatt, J., Harris, R., and Yao, X. (2005). Diversity creation methods: a survey and categorisation. *Information Fusion*, 6(1):5–20.

[11] Bryll, R., Gutierrez-Osuna, R., and Quek, F. (2003). Attribute bagging: improving accuracy of classifier ensembles by using random feature subsets. *Pattern recognition*, 36(6):1291–1302.

[12] Burger, T., Aran, O., and Caplier, A. (2006). Modeling hesitation and conflict: a belief-based approach for multi-class problems. In *Machine Learning and Applications, 2006. ICMLA'06. 5th International Conference on*, pages 95–100. IEEE.

[13] Casella, G. and Berger, R. L. (2002). *Statistical inference*, volume 2. Duxbury Pacific Grove, CA.

[14] Cho, S.-B. and Kim, J. H. (1995). Combining multiple neural networks by fuzzy integral for robust classification. *IEEE Transactions on Systems, Man, and Cybernetics*, 25(2):380–384.

[15] Clemen, R. T. and Winkler, R. L. (1999). Combining probability distributions from experts in risk analysis. *Risk analysis*, 19(2):187–203.

[16] Cunningham, P. and Carney, J. (2000). Diversity versus quality in classification ensembles based on feature selection. In *European Conference on Machine Learning*, pages 109–116. Springer.

[17] Dawid, A. P. and Skene, A. M. (1979a). Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied statistics*, pages 20–28.

[18] Dawid, A. P. and Skene, A. M. (1979b). Maximum likelihood estimation of observer error-rates using the EM algorithm. *Applied statistics*, pages 20–28.

[19] De Maesschalck, R., Jouan-Rimbaud, D., and Massart, D. L. (2000). The mahalanobis distance. *Chemometrics and intelligent laboratory systems*, 50(1):1–18.

[20] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391.

[21] Dempster, A. P. (1967). Upper and lower probabilities induced by a multiple valued mapping. *Annals of Mathematical Satistics*, 38(2):325–339.

[22] Deng, X., Liu, Q., Deng, Y., and Mahadevan, S. (2016). An improved method to construct basic probability assignment based on the confusion matrix for classification problem. *Information Sciences*, 340:250–261.

[23] Detyniecki, M., Yager, R. R., and Bouchon-Meunier, B. (2002). Reducing t-norms and augmenting t-conorms. *International Journal of General Systems*, 31(3):265–276.

[24] Dietterich, T. G. (2000). Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer.

[25] Dietterich, T. G. and Bakiri, G. (1995a). Solving multiclass learning problems via error-correcting output codes. *Journal of artificial intelligence research*, 2:263–286.

[26] Dietterich, T. G. and Bakiri, G. (1995b). Solving multiclass learning problems via error-correcting output codes. *Journal of artificial intelligence research*, 2:263–286.

[27] Dubois, D., Foulloy, L., Mauris, G., and Prade, H. (2004). Probability-possibility transformations, triangular fuzzy sets, and probabilistic inequalities. *Reliable computing*, 10(4):273–297.

[28] Dubois, D. and Prade, H. (1982). On several representations of an uncertain body of evidence. *Fuzzy Information and Decision Processes*, pages 161–181.

[29] Dubois, D. and Prade, H. (1988). *Possibility theory: An approach to the computerized processing of information*. Plenum Press, New York.

[30] Dubois, D. and Prade, H. (2015). *Possibility Theory and Its Applications: Where Do We Stand?*, pages 31–60. Springer Berlin Heidelberg, Berlin, Heidelberg.

[31] Džeroski, S. and Ženko, B. (2004). Is combining classifiers with stacking better than selecting the best one? *Machine learning*, 54(3):255–273.

[32] Faugeras, O. P. (2017). Inference for copula modeling of discrete data: a cautionary tale and some facts. *Dependence Modeling*, 5(1):121–132.

[33] Fishburn, P. (1999). Preference structures and their numerical representations. *Theoretical Computer Science*, 217(2):359–383.

[34] Fleiss, J. L. and Cuzick, J. (1979). The reliability of dichotomous judgments: Unequal numbers of judges per subject. *Applied Psychological Measurement*, 3(4):537–542.

[35] French, S. (1983). *Group consensus probability distributions: A critical survey*. University of Manchester. Department of Decision Theory.

[36] Freund, Y. and Schapire, R. E. (1996). Experiments with a new boosting algorithm. In Saitta, L., editor, *Proceedings of the Thirteenth International Conference on Machine Learning (ICML 1996)*, pages 148–156. Morgan Kaufmann.

[37] Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139.

[38] Gader, P. D., Mohamed, M. A., and Keller, J. M. (1996). Fusion of handwritten word classifiers. *Pattern Recognition Letters*, 17(6):577–584.

[39] Garcia Marquez, C. (2014). Multivariate kernel functions for categorical variables.

[40] Genest, C. and Nešlehová, J. (2007). A primer on copulas for count data. *ASTIN Bulletin: The Journal of the International Actuarial Association*, 37(2):475–515.

[41] Giacinto, G. and Roli, F. (2001). Design of effective neural network ensembles for image classification purposes. *Image and Vision Computing*, 19(9-10):699–707.

[42] Gilboa, I. (1987). Expected utility with purely subjective non-additive probabilities. *Journal of Mathematical Economics*, 16(1):65 – 88.

[43] Giompapa, S., Farina, A., Gini, F., Graziano, A., Croci, R., and Di Stefano, R. (2008). Naval target classification based on the confusion matrix. In *Aerospace Conference, 2008 IEEE*, pages 1–9. IEEE.

[44] Hansen, L. K. and Salamon, P. (1990). Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*, 12(10):993–1001.

[45] Heskes, T. (1998). Selecting weighting factors in logarithmic opinion pools. In *Advances in neural information processing systems*, pages 266–272.

[46] Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence*, 20(8):832–844.

[47] Ho, T. K., Hull, J. J., and Srihari, S. N. (1992). On multiple classifier systems for pattern recognition. In *Pattern Recognition, 1992. Vol. II. Conference B: Pattern Recognition Methodology and Systems, Proceedings., 11th IAPR International Conference on*, pages 84–87. IEEE.

[48] Holmes, C. and Adams, N. (2002). A probabilistic nearest neighbour method for statistical pattern recognition. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 64(2):295–306.

[49] Hong, C., Batal, I., and Hauskrecht, M. (2014). A mixtures-of-experts framework for multi-label classification. *arXiv preprint arXiv:1409.4698*.

[50] Hong, J.-H., Min, J.-K., Cho, U.-K., and Cho, S.-B. (2008). Fingerprint classification using one-vs-all support vector machines dynamically ordered with naı¨ ve bayes classifiers. *Pattern Recognition*, 41(2):662–671.

[51] Huang, Y. S. and Suen, C. Y. (1993). The behavior-knowledge space method for combination of multiple classifiers. In *IEEE computer society conference on computer vision and pattern recognition*, pages 347–347. Institute of Electrical Engineers Inc (IEEE).

[52] Huang, Y. S. and Suen, C. Y. (1995). A method of combining multiple experts for the recognition of unconstrained handwritten numerals. *IEEE transactions on pattern analysis and machine intelligence*, 17(1):90–94.

[53] Jackowski, K. and Wozniak, M. (2009). Algorithm of designing compound recognition system on the basis of combining classifiers with simultaneous splitting feature space into competence areas. *Pattern Analysis and Applications*, 12(4):415.

[54] Jeong, S., Lim, K.-T., and Nam, Y.-S. (2002). A combination method of two classifiers based on the information of confusion matrix. In *Frontiers in Handwriting Recognition, 2002. Proceedings. Eighth International Workshop on*, pages 519–523. IEEE.

[55] Kearns, M. and Valiant, L. (1994). Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM (JACM)*, 41(1):67–95.

[56] Keller, J. M., Gader, P., Tahani, H., Chiang, J.-H., and Mohamed, M. (1994). Advances in fuzzy integration for pattern recognition. *Fuzzy Sets and Systems*, 65(2):273 – 283. Fuzzy Methods for Computer Vision and Pattern Recognition.

[57] Kim, H.-C. and Ghahramani, Z. (2012). Bayesian classifier combination. In Lawrence, N. D. and Girolami, M., editors, *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, volume 22 of *Proceedings of Machine Learning Research*, pages 619–627, La Palma, Canary Islands. PMLR.

[58] Kim, J. H., Kim, K. K., Nadal, C. P., and Suen, C. Y. (2000). A methodology of combining hmm and mlp classifiers for cursive word recognition. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 2, pages 319–322. IEEE.

[59] Kittler, J., Ahmadyfard, A., and Windridge, D. (2003). Serial multiple classifier systems exploiting a coarse to fine output coding. In *International Workshop on Multiple Classifier Systems*, pages 106–114. Springer.

[60] Kittler, J., Hatef, M., Duin, R. P., and Matas, J. (1998). On combining classifiers. *IEEE transactions on pattern analysis and machine intelligence*, 20(3):226–239.

[61] Ko, A. H., Sabourin, R., and Britto Jr, A. S. (2008). From dynamic classifier selection to dynamic ensemble selection. *Pattern Recognition*, 41(5):1718–1731.

[62] Kohavi, R., Wolpert, D. H., et al. (1996). Bias plus variance decomposition for zero-one loss functions. In *ICML*, volume 96, pages 275–83.

[63] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[64] Krogh, A. and Vedelsby, J. (1995). Neural network ensembles, cross validation, and active learning. In *Advances in neural information processing systems*, pages 231–238.

[65] Kuncheva, L. I. (2000). Clustering-and-selection model for classifier combination. In *Knowledge-Based Intelligent Engineering Systems and Allied Technologies, 2000. Proceedings. Fourth International Conference on*, volume 1, pages 185–188. IEEE.

[66] Kuncheva, L. I. (2003). " fuzzy" versus" nonfuzzy" in combining classifiers designed by boosting. *IEEE Transactions on fuzzy systems*, 11(6):729–741.

[67] Kuncheva, L. I. (2004). *Combining pattern classifiers: methods and algorithms*. John Wiley & Sons.

[68] Kuncheva, L. I., Bezdek, J. C., and Duin, R. P. (2001). Decision templates for multiple classifier fusion: an experimental comparison. *Pattern recognition*, 34(2):299–314.

[69] Kuncheva, L. I. and Whitaker, C. J. (2003). Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine learning*, 51(2):181–207.

[70] Lacoste, A., Marchand, M., Laviolette, F., and Larochelle, H. (2014). Agnostic bayesian learning of ensembles. In Jebara, T. and Xing, E. P., editors, *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 611–619. JMLR Workshop and Conference Proceedings.

[71] Laplace, P.-S. (1840). *Essai philosophique sur les probabilités*. Bachelier.

[72] Lecué, G. et al. (2007). Optimal rates of aggregation in classification under low noise assumption. *Bernoulli*, 13(4):1000–1022.

[73] Lee, P. M. (2012). *Bayesian statistics: an introduction*. John Wiley & Sons.

[74] Louppe, G. (2014). Understanding random forests: From theory to practice. *arXiv preprint arXiv:1407.7502*.

[75] Mallah, C. D. and Orwell, J. (2013). Probabilistic classification from a k-nearest-neighbour classifier. *Computational Research*, 1(1):1–9.

[76] Margineantu, D. D. and Dietterich, T. G. (1997). Pruning adaptive boosting. In *ICML*, volume 97, pages 211–218.

[77] Masoudnia, S. and Ebrahimpour, R. (2014). Mixture of experts: a literature survey. *Artificial Intelligence Review*, 42(2):275–293.

[78] Matan, O. (1996). On voting ensembles of classifiers. In *Proceedings of AAAI-96 workshop on integrating multiple learned models*, pages 84–88. Citeseer.

[79] Mayhua-López, E., Gómez-Verdejo, V., and Figueiras-Vidal, A. R. (2015). A new boosting design of support vector machine classifiers. *Information Fusion*, 25:63–71.

[80] Parker, J. (2001). Rank and response combination from confusion matrix data. *Information fusion*, 2(2):113–120.

[81] Quost, B., Masson, M.-H., and Denœux, T. (2011). Classifier fusion in the dempster–shafer framework using optimized t-norm based combination rules. *International Journal of Approximate Reasoning*, 52(3):353–374.

[82] Ranjan, R. and Gneiting, T. (2010). Combining probability forecasts. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(1):71–91.

[83] Reformat, M. and Yager, R. R. (2008). Building ensemble classifiers using belief functions and owa operators. *Soft Computing*, 12(6):543–558.

[84] Rogova, G. (1994). Combining the results of several neural network classifiers. *Neural Networks*, 7(5):777 – 781.

[85] Schapire, R. E. (1990). The strength of weak learnability. *Machine learning*, 5(2):197–227.

[86] Seewald, A. K. (2002). Meta-learning for stacked classification. *audiology*, 24(226):69.

[87] Shafer, G. (1976). *A Mathematical Theory of Evidence*. Princeton University press, Princeton (NJ), USA.

[88] Shipp, C. A. and Kuncheva, L. I. (2002). Relationships between combination methods and measures of diversity in combining classifiers. *Information fusion*, 3(2):135–148.

[89] Skalak, D. B. et al. (1996). The sources of increased accuracy for two proposed boosting algorithms. In *Proc. American Association for Artificial Intelligence, AAAI-96, Integrating Multiple Learned Models Workshop*, volume 1129, page 1133. Citeseer.

[90] Sklar, A. (1959). Fonctions de répartition à n dimensions et leurs marges. publications de l'institut de statistique de l'université de paris.

[91] Skurichina, M. (2001). Stabilizing weak classifiers: Regularization and combining techniques in discriminant analysis.

[92] Sontag, E. D. (1998). A learning result for continuous-time recurrent neural networks1. *Systems & control letters*, 34(3):151–158.

[93] Sutton, R. S. and Barto, A. G. (1998). *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

[94] Tax, D. M. and Duin, R. P. (2001). Combining one-class classifiers. In *International Workshop on Multiple Classifier Systems*, pages 299–308. Springer.

[95] Ting, K. M. and Witten, I. H. (1999). Issues in stacked generalization. *J. Artif. Intell. Res.(JAIR)*, 10:271–289.

[96] Todorovski, L. and Džeroski, S. (2003). Combining classifiers with meta decision trees. *Machine learning*, 50(3):223–249.

[97] Tomašev, N., Radovanovic, M., Mladenic, D., and Ivanovic, M. (2011). A probabilistic approach to nearest-neighbor classification: Naive hubness bayesian knn. In *Proc. 20th ACM Int. Conf. on Information and Knowledge Management (CIKM)*, pages 2173–2176.

[98] Wolpert, D. H. (1992). Stacked generalization. *Neural networks*, 5(2):241–259.

[99] Wolpert, D. H. (1996). The lack of a priori distinctions between learning algorithms. *Neural computation*, 8(7):1341–1390.

[100] Woźniak, M., Graña, M., and Corchado, E. (2014). A survey of multiple classifier systems as hybrid systems. *Information Fusion*, 16:3–17.

[101] Xu, L., Krzyzak, A., and Suen, C. Y. (1992). Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Transactions on systems, man, and cybernetics*, 22(3):418–435.

[102] Zadeh, L. (1978). A., 1978. fuzzy sets as a basis for a theory of possibility. *Fuzzy sets and systems*, 1(1):3–28.

[103] Zadeh, L. A. (1965). Information and control. *Fuzzy sets*, 8(3):338–353.

[104] Zadeh, L. A. (1996). Fuzzy sets. In *Fuzzy Sets, Fuzzy Logic, And Fuzzy Systems: Selected Papers by Lotfi A Zadeh*, pages 394–432. World Scientific.

[105] Zadeh, L. A. (1999). Fuzzy sets as a basis for a theory of possibility. *Fuzzy sets and systems*, 100:9–34.

[106] Zezula, I. (2009). On multivariate gaussian copulas. *Journal of Statistical Planning and Inference*, 139(11):3942 – 3946. Special Issue: The 8th Tartu Conference on Multivariate Statistics & The 6th Conference on Multivariate Distributions with Fixed Marginals.

[107] Zhang, Z., Luo, P., Loy, C. C., and Tang, X. (2014). *Facial Landmark Detection by Deep Multi-task Learning*, pages 94–108. Springer International Publishing, Cham.

Appendix A

Families of *t*-norms

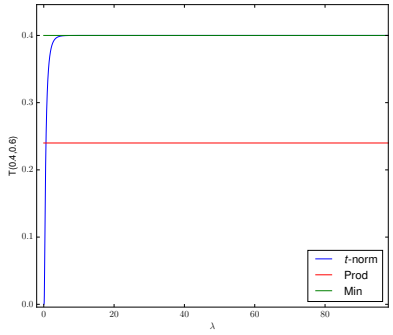
This appendix is a non exhaustive list of *t*-norm families. Since we are interested in parametrized *t*-norms, we only list those that depend on a parameter. Besides Aczel-Alsina,

T-norm	Formula	Parameter interval
Schweizer–Sklar $\mathrm{T}_\lambda(a,b)$	$\min(a,b)$ $(a^\lambda+b^\lambda-1)^{\frac{1}{\lambda}}$ $a.b$ $\max(0,(a^\lambda+b^\lambda-1)^{\frac{1}{\lambda}})$ $\mathrm{T}^{\mathrm{D}}(a,b)$	$\lambda=-\infty$ $-\infty<\lambda<0$ $\lambda=0$ $0<\lambda<+\infty$ $\lambda=+\infty$
Frank $\mathrm{T}_\lambda(a,b)$	$\min(a,b)$ $a.b$ $\mathrm{T}^{\mathrm{Luk}}(a,b)$ $\log_\lambda\left(1+\frac{(\lambda^a-1).(\lambda^b-1)}{\lambda+1}\right)$	$\lambda=0$ $\lambda=1$ $\lambda=+\infty$ $\lambda\in]0;+\infty[/\{1\}$
Dombi $\mathrm{T}_\lambda(a,b)$	$\mathrm{T}^{\mathrm{D}}(a,b)$ $\min(a,b)$ $\frac{1}{1+\left((\frac{1-a}{a})^\lambda+(\frac{1-b}{b})^\lambda\right)^{\frac{1}{\lambda}}}$	$\lambda=0$ $\lambda=+\infty$ $0<\lambda<+\infty$
Yager $\mathrm{T}_\lambda(a,b)$	$\mathrm{T}^{\mathrm{D}}(a,b)$ $\max(0,1-((1-a)^\lambda+(1-b)^\lambda)^{\frac{1}{\lambda}})$ $\min(a,b)$	$\lambda=0$ $0<\lambda<+\infty$ $\lambda=+\infty$

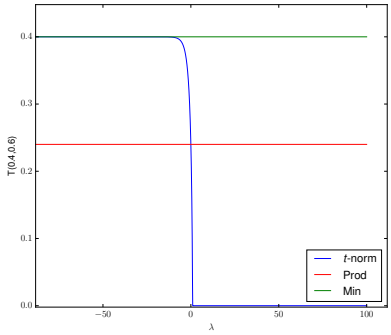
Table A.1 Specifications of *t*-norms families.

we present four parametrized families of *t*-norms : Schweizer–Sklar, Frank, Dombi and Yager *t*-norm families. Their mathematical formulas as well as the interval of their parameters are

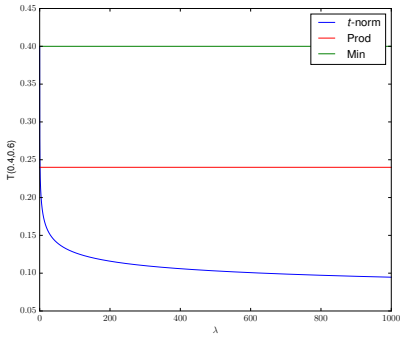
presented in Table A.1. Note that $\mathrm{T}^{D}(a,b)$ is called the Drastic t-norm (if $a=1,\mathrm{T}(a,b)=b$ and if $b=1,\mathrm{T}(a,b)=a$ otherwise $\mathrm{T}(a,b)=0$) and $\mathrm{T}^{\mathrm{Luk}}(a,b)$ is called the Lukasiewicz t-norm ($\mathrm{T}(a,b)=\max(0,a+b-1)$).



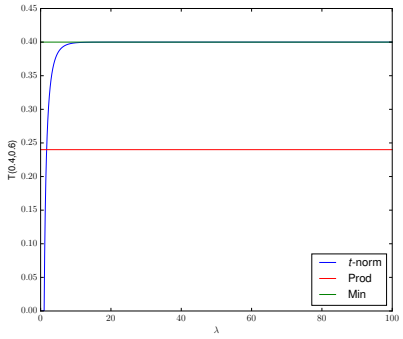
(a) Dombi



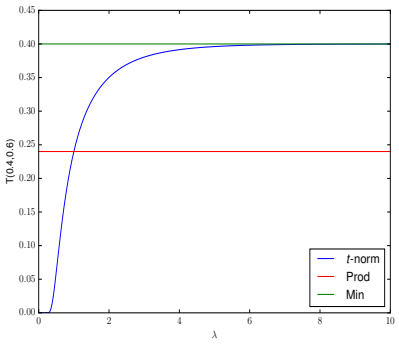
(b) Schweizer–Sklar



(c) Frank



(d) Yager



(e) Aczel-Alsina

Fig. A.1 Plots of different t-norm families for a fixed input pair (0.4,0.6).

As indicated in section 2.3.2, the differences in performances between different families of t-norms are not significant. This is actually due to the fact that they are parametrized.

Indeed, given a large grid of the parameter λ, the range of the outputs $\{\mathrm{T}_\lambda(a,b)\}_{grid_\lambda}$ of an arbitrary pair of inputs (a,b) is highly overlapping across different families of t-norm. For instance, let us consider the pair (0.4,0.6) and plot $\mathrm{T}_\lambda(a,b)$ for the four t-norms rule in the previous table. The plots are presented in Figure A.1. We obviously see that except for Frank family, other t-norms outputs belong to the interval $[0.0,0.4]$ whose bounds correspond to the drastic and the minimum rule which are the ultimate smallest and the ultimate largest t-norms [23]. Besides the minimum and the drastic rules are reachable for Schweizer–Sklar, Dombi and Yager families for every $(a,b) \in [0,1]^2$. However, for Frank family, the bounds are the Lukasiewicz and the minimum t-norms. Thus, its range of outputs $\{\mathrm{T}_\lambda(a,b)\}_{grid_\lambda}$ is slightly smaller than the one of other families. In conclusion, the usage of Aczel-Alsina, Schweizer–Sklar, Dombi or Yager families will induce close performances since all values between the drastic and the minimum rule are reachable for all pairs of inputs. In theory, for Frank family, the performances are expected to be different. In practice, we have run similar experiments as in chapter 2 using Franck t-norm and slightly poorer results were observed.

Appendix B

Detailed Wilcoxon signed-rank test results for *t*-norm approach

This appendix gives detailed statistical test results for the statistical significance analyses carried out in 2.4.6. In 2.4.5, we combined homogeneously three different types of classifiers for 7 datasets therefore we obtain two series of 21 accuracies for each pair of compared methods. The green check mark in the tables indicates that the test is passed while the red cross indicates the opposite. If the test is passed, we can look for the best method in tables 2.3, 2.4 and 2.5, otherwise the result is undecided. Each of the following table contains 5 columns: name of the dataset, the value of the statistic, the *p*-value, the conclusion of the test (i.e. passed or not) and the outperforming classification method if the test is passed. Two approaches may have tied performances even though the test is passed because a distribution can be asymmetric and have a null expectation. So an experiment is really conclusive when the test is passed and expected accuracies are different.

Dataset	stat	*p*-value	CCl	Best method
Digits	994.0	0.0	✓	Multiplicative
Waveform	1941.5	0.35	✗	Undecided
Cancer	1373.0	0.0	✓	Multiplicative
CNAE	1567.5	0.0009	✓	*t*-norm
Segments	2312.0	0.46	✗	Undecided
Wine	161.0	0.0	✓	Multiplicative
Mnist	2426.0	0.86	✗	Undecided

Table B.1 Wilcoxon test for *t*-norm and multiplicative rule. Classifiers are decision trees.

Dataset	stat	p-value	CCl	Best method
Digits	2.0	0.0	✓	t-norm
Waveform	0.0	0.0	✓	t-norm
Cancer	757.0	0.0	✓	t-norm
CNAE	889.0	0.0	✓	t-norm
Segments	1384.0	0.0	✓	t-norm
Wine	437.0	0.0	✓	t-norm
Mnist	2.0	0.0	✓	t-norm

Table B.2 Wilcoxon test for t-norm and minimum rule. Classifiers are decision trees.

Dataset	stat	p-value	CCl	Best method
Digits	1.0	0.0	✓	Bayes
Waveform	1122.0	0.0	✓	Bayes
Cancer	115.0	0.0	✓	Bayes
CNAE	0.0	0.0	✓	Bayes
Segments	371.0	0.0	✓	Bayes
Wine	109.0	0.0	✓	Bayes
Mnist	0.0	0.0	✓	Bayes

Table B.3 Wilcoxon test for t-norm and Bayes rule. Classifiers are decision trees.

Dataset	stat	p-value	CCl	Best method
Digits	0.0	0.0	✓	t-norm
Waveform	0.0	0.0	✓	t-norm
Cancer	63.0	0.0	✓	t-norm
CNAE	9.0	0.0	✓	t-norm
Segments	14.0	0.0	✓	t-norm
Wine	306.0	0.0	✓	t-norm
Mnist	0.0	0.0	✓	t-norm

Table B.4 Wilcoxon test for t-norm and stacking rule. Classifiers are decision trees.

Dataset	stat	p-value	CCl	Best method
Digits	0.0	0.0	✓	t-norm
Waveform	1223.0	0.0	✓	t-norm
Cancer	680.0	0.0	✓	t-norm
CNAE	1.0	0.0	✓	t-norm
Segments	0.0	0.0	✓	t-norm
Wine	404.0	0.0	✓	Weighted vote
Mnist	0.0	0.0	✓	t-norm

Table B.5 Wilcoxon test for t-norm and weighted vote. Classifiers are decision trees.

Dataset	stat	p-value	CCl	Best method
Digits	0.0	0.0	✓	t-norm
Waveform	0.0	0.0	✓	t-norm
Cancer	122.0	0.0	✓	t-norm
CNAE	0.0	0.0	✓	t-norm
Segments	82.0	0.0	✓	t-norm
Wine	1407.0	0.00012	✓	t-norm
Mnist	0.0	0.0	✓	t-norm

Table B.6 Wilcoxon test for t-norm and classifier selection. Classifiers are decision trees.

Dataset	stat	p-value	CCl	Best method
Digits	0.0	0.0	✓	t-norm
Waveform	0.0	0.0	✓	t-norm
Cancer	392.0	0.0	✓	t-norm
CNAE	1.0	0.0	✓	t-norm
Segments	82.0	0.0	✓	t-norm
Wine	1902.0	0.03	✓	Max. Ind. Clf.
Mnist	0.0	0.0	✓	t-norm

Table B.7 Wilcoxon test for t-norm and maximally accurate individual classifier. Classifiers are decision trees.

Dataset	stat	p-value	CCl	Best method
Digits	0.0	0.0	✓	t-norm
Waveform	0.0	0.0	✓	t-norm
Cancer	225.0	0.0	✓	t-norm
CNAE	1.0	0.0	✓	t-norm
Segments	0.0	0.0	✓	t-norm
Wine	2091.0	0.135	✗	Undecided
Mnist	0.0	0.0	✓	t-norm

Table B.8 Wilcoxon test for t-norm and centralized classifier. Classifiers are decision trees.

Dataset	stat	p-value	CCl	Best method
Digits	17.0	0.0	✓	Multiplicative
Waveform	902.5	0.046	✓	Multiplicative
Cancer	1.0	0.0	✓	tie
CNAE	1131.5	0.025	✓	Multiplicative
Segments	2390.0	0.64	✗	Undecided
Wine	1594.0	0.00095	✓	Multiplicative
Mnist	0.0	0.0	✓	t-norm

Table B.9 Wilcoxon test for t-norm and multiplicative rule. Classifiers are logistic regressions.

Dataset	stat	p-value	CCl	Best method
Digits	0.0	0.0	✓	t-norm
Waveform	0.0	0.0	✓	t-norm
Cancer	629.0	0.0	✓	t-norm
CNAE	0.0	0.0	✓	t-norm
Segments	142.0	0.0	✓	t-norm
Wine	1301.0	0.0	✓	t-norm
Mnist	0.0	0.0	✓	t-norm

Table B.10 Wilcoxon test for t-norm and minimum rule. Classifiers are logistic regressions.

Dataset	stat	p-value	CCl	Best method
Digits	36.0	0.0	✓	Bayes
Waveform	759.5	0.0	✓	Bayes
Cancer	1.0	0.0	✓	tie
CNAE	0.0	0.0	✓	Bayes
Segments	923.0	0.0	✓	Bayes
Wine	510.0	0.0	✓	Bayes
Mnist	0.0	0.0	✓	t-norm

Table B.11 Wilcoxon test for t-norm and Bayes rule. Classifiers are logistic regressions.

Dataset	stat	p-value	CCl	Best method
Digits	0.0	0.0	✓	t-norm
Waveform	0.0	0.0	✓	t-norm
Cancer	0.0	0.0	✓	t-norm
CNAE	0.0	0.0	✓	t-norm
Segments	0.0	0.0	✓	t-norm
Wine	0.0	0.0	✓	t-norm
Mnist	0.0	0.0	✓	t-norm

Table B.12 Wilcoxon test for t-norm and stacking. Classifiers are logistic regressions.

Dataset	stat	p-value	CCl	Best method
Digits	1477.0	0.0003	✓	t-norm
Waveform	1326.0	0.74	✗	Undecided
Cancer	1.0	0.0	✓	tie
CNAE	65.0	0.0	✓	t-norm
Segments	0.0	0.0	✓	t-norm
Wine	2306.0	0.451	✗	Undecided
Mnist	0.0	0.0	✓	t-norm

Table B.13 Wilcoxon test for t-norm and weighted vote. Classifiers are logistic regressions.

Dataset	stat	p-value	CCl	Best method
Digits	47.0	0.0	✓	t-norm
Waveform	0.0	0.0	✓	t-norm
Cancer	53.5	0.0	✓	tie
CNAE	0.0	0.0	✓	t-norm
Segments	79.0	0.0	✓	t-norm
Wine	1089.0	0.0	✓	t-norm
Mnist	0.0	0.0	✓	t-norm

Table B.14 Wilcoxon test for t-norm and classifier selection. Classifiers are logistic regressions.

Dataset	stat	p-value	CCl	Best method
Digits	74.0	0.0	✓	t-norm
Waveform	0.0	0.0	✓	t-norm
Cancer	1.0	0.0	✓	Max. Ind. Clf.
CNAE	0.0	0.0	✓	t-norm
Segments	79.0	0.0	✓	t-norm
Wine	2493.0	0.912	✗	Undecided
Mnist	0.0	0.0	✓	t-norm

Table B.15 Wilcoxon test for t-norm and maximally accurate individual classifier. Classifiers are logistic regressions.

Dataset	stat	p-value	CCl	Best method
Digits	0.0	0.0	✓	Centralized
Waveform	0.0	0.0	✓	Centralized
Cancer	1581.0	0.212	✗	Undecided
CNAE	0.0	0.0	✓	Centralized
Segments	108.0	0.0	✓	Centralized
Wine	1.0	0.0	✓	Centralized
Mnist	0.0	0.0	✓	Centralized

Table B.16 Wilcoxon test for t-norm and centralized classifier. Classifiers are logistic regressions.

Dataset	stat	p-value	CCl	Best method
Digits	874.0	0.0	✓	Multiplicative
Waveform	600.0	0.0	✓	Multiplicative
Cancer	732.0	0.0	✓	Multiplicative
CNAE	901.0	0.416	✗	Undecided
Segments	1391.0	0.0	✓	t-norm
Wine	401.0	0.0	✓	Multiplicative
Mnist	1378.0	0.0	✓	t-norm

Table B.17 Wilcoxon test for t-norm and multiplicative rule. Classifiers are Gaussian naive Bayesian.

Dataset	stat	p-value	CCl	Best method
Digits	0.0	0.0	✓	t-norm
Waveform	1.0	0.0	✓	t-norm
Cancer	6.0	0.0	✓	t-norm
CNAE	0.0	0.0	✓	t-norm
Segments	242.0	0.0	✓	t-norm
Wine	460.0	0.0	✓	t-norm
Mnist	0.0	0.0	✓	t-norm

Table B.18 Wilcoxon test for t-norm and minimum rule. Classifiers are Gaussian naive Bayesian.

Dataset	stat	p-value	CCl	Best method
Digits	0.0	0.0	✓	Bayes
Waveform	752.0	0.0	✓	Bayes
Cancer	0.0	0.0	✓	Bayes
CNAE	0.0	0.0	✓	Bayes
Segments	242.0	0.0	✓	Bayes
Wine	297.0	0.0	✓	Bayes
Mnist	0.0	0.0	✓	Bayes

Table B.19 Wilcoxon test for t-norm and Bayes rule. Classifiers are Gaussian naive Bayesian.

Dataset	stat	p-value	CCl	Best method
Digits	0.0	0.0	✓	t-norm
Waveform	104.5	0.0	✓	t-norm
Cancer	3.0	0.0	✓	t-norm
CNAE	0.0	0.0	✓	t-norm
Segments	4.0	0.0	✓	t-norm
Wine	1030.0	0.0	✓	t-norm
Mnist	0.0	0.0	✓	t-norm

Table B.20 Wilcoxon test for t-norm and stacking. Classifiers are Gaussian naive Bayesian.

Dataset	stat	p-value	CCl	Best method
Digits	0.0	0.0	✓	t-norm
Waveform	72.0	0.0	✓	t-norm
Cancer	1878.0	0.026	✓	Weighted vote
CNAE	7.0	0.0	✓	t-norm
Segments	0.0	0.0	✓	t-norm
Wine	511.0	0.0	✓	Weighted vote
Mnist	0.0	0.0	✓	t-norm

Table B.21 Wilcoxon test for t-norm and weighted vote. Classifiers are Gaussian naive Bayesian.

Dataset	stat	p-value	CCl	Best method
Digits	1.0	0.0	✓	t-norm
Waveform	19.0	0.0	✓	t-norm
Cancer	13.0	0.0	✓	t-norm
CNAE	0.0	0.0	✓	t-norm
Segments	46.0	0.0	✓	t-norm
Wine	155.0	0.0	✓	t-norm
Mnist	0.0	0.0	✓	t-norm

Table B.22 Wilcoxon test for t-norm and classifier selection. Classifiers are Gaussian naive Bayesian.

Dataset	stat	p-value	CCl	Best method
Digits	0.0	0.0	✓	t-norm
Waveform	19.0	0.0	✓	t-norm
Cancer	57.0	0.0	✓	t-norm
CNAE	0.0	0.0	✓	t-norm
Segments	48.0	0.0	✓	t-norm
Wine	1405.0	0.00011	✓	Max. Ind. Clf.
Mnist	0.0	0.0	✓	t-norm

Table B.23 Wilcoxon test for t-norm and maximally accurate individual classifier. Classifiers are Gaussian naive Bayesian.

Dataset	stat	*p*-value	CCl	Best method
Digits	44.0	0.0	✓	Centralized
Waveform	29.0	0.0	✓	Centralized
Cancer	659.0	0.0	✓	*t*-norm
CNAE	0.0	0.0	✓	Centralized
Segments	16.0	0.0	✓	*t*-norm
Wine	22.0	0.0	✓	Centralized
Mnist	0.0	0.0	✓	*t*-norm

Table B.24 Wilcoxon test for *t*-norm and centralized classifier. Classifiers are Gaussian naive Bayesian.

Appendix C

Hyperparameters of base classifiers

This appendix gives a few implementation details for the reproducibility of experiments presented in section 2.4. We used classifier implementation from the python Scikit-learn library 0.17.1. Except for the maximal depth parameter of decision trees which is set to 3, all other parameters are left to default values. Note that we use a naive Bayes classifier with Gaussian class conditional densities. More precisely, the classifiers are instantiated as :

- decision tree : `sklearn.tree.DecisionTreeClassifier(max_depth = 3)`, the split criterion is Gini impurity (CART algorithm),
- logistic regression : `sklearn.linear_model.LogisticRegression()`, the the regularization parameter is 1 and the corresponding penalty is in norm L_2,
- naive Bayes : `sklearn.naive_bayes.GaussianNB()`, class probabilities are ML estimates.

When best accuracy is sought for a given dataset, each hyperparameter should preferably be tuned using a cross validated grid search as we do for λ but to assess combination method performances, we argue that it is preferable to avoid tweaking these parameters too much as the conclusions would be dependent on the method used to that end. In addition, the fusion should be efficient regardless of base classifier hyperparameter values.

Appendix D

Detailed Wilcoxon signed-rank test results for copula approach

This appendix gives detailed statistical test results for the statistical significance analyses carried out in 3.5.4. In 3.5.2 and 3.5.3, we combined homogeneously three logistic regression classifiers for 3 synthetic and 3 real datasets with different number of training samples and different numbers of base classifiers therefore we obtain two series of 15 accuracies for each pair of compared methods.

The green check mark in the tables indicates that the test is passed while the red cross indicates the opposite. If the test is passed, we can look for the best method in tables 3.1, 3.2 for synthetic datasets and 3.4, 3.5 and 3.6 for real datasets., otherwise the result is undecided. Each of the following table contains 5 columns: name of the dataset, the value of the statistic, the p-value, the conclusion of the test (i.e. passed or not) and the outperforming classification method if the test is passed. Two approaches may have tied performances even though the test is passed because a distribution can be asymmetric and have a null expectation. So an experiment is really conclusive when the test is passed and expected accuracies are different. The wilcoxon results for the real dataset are presented in section D.1 and those for synthetic datasets are presented in section D.2.

D.1 Wilcoxon results for synthetic dataset

Dataset	stat	p-value	CCl	Best method
Blobs ($N_{\text{train}} = 200$)	866659	0.0438	✓	Gaussian copula
Blobs ($N_{\text{train}} = 400$)	431483	0.0	✓	Gaussian copula
Circles ($N_{\text{train}} = 200$)	0.0	0.0	✓	Gaussian copula
Circles ($N_{\text{train}} = 400$)	0.0	0.0	✓	Gaussian copula
Moons ($N_{\text{train}} = 200$)	113161.5	0.0	✓	Independent copula
Moons ($N_{\text{train}} = 400$)	56518	0.0	✓	Gaussian copula

Table D.1 Wilcoxon test for Gaussian and independent copulas-based models. Classifiers are logistic regressions.

Dataset	stat	p-value	CCl	Best method
Blobs ($N_{\text{train}} = 200$)	599	0.0	✓	Gaussian copula
Blobs ($N_{\text{train}} = 400$)	0.0	0.0	✓	Gaussian copula
Circles ($N_{\text{train}} = 200$)	0.0	0.0	✓	Gaussian copula
Circles ($N_{\text{train}} = 400$)	0.0	0.0	✓	Gaussian copula
Moons ($N_{\text{train}} = 200$)	902066	0.0	✓	Stacking
Moons ($N_{\text{train}} = 400$)	34017	0.0	✓	Gaussian copula

Table D.2 Wilcoxon test for Gaussian copulas-based model and stacking. Classifiers are logistic regressions.

Dataset	stat	p-value	CCl	Best method
Blobs ($N_{\text{train}} = 200$)	2602	0.0	✓	Gaussian copula
Blobs ($N_{\text{train}} = 400$)	0.0	0.0	✓	Gaussian copula
Circles ($N_{\text{train}} = 200$)	2983	0.0	✓	Gaussian copula
Circles ($N_{\text{train}} = 400$)	0.0	0.0	✓	Gaussian copula
Moons ($N_{\text{train}} = 200$)	1084254.5	0.0	✓	Gaussian copula
Moons ($N_{\text{train}} = 400$)	16294	0.0	✓	Gaussian copula

Table D.3 Wilcoxon test for Gaussian copulas-based model and classifier selection. Classifiers are logistic regressions.

Dataset	stat	p-value	CCl	Best method
Blobs ($N_{\text{train}} = 200$)	88239.5	0.0	✓	Gaussian copula
Blobs ($N_{\text{train}} = 400$)	2362	0.0	✓	Gaussian copula
Circles ($N_{\text{train}} = 200$)	23	0.0	✓	Gaussian copula
Circles ($N_{\text{train}} = 400$)	0.0	0.0	✓	Gaussian copula
Moons ($N_{\text{train}} = 200$)	3500.0	0.0	✓	Weighted vote
Moons ($N_{\text{train}} = 400$)	6	0.0	✓	Weighted vote

Table D.4 Wilcoxon test for Gaussian copulas-based model and weighted vote. Classifiers are logistic regressions.

Dataset	stat	p-value	CCl	Best method
Blobs ($N_{\text{train}} = 200$)	430415	0.0	✓	Gaussian copula
Blobs ($N_{\text{train}} = 400$)	20936	0.0	✓	Gaussian copula
Circles ($N_{\text{train}} = 200$)	10.0	0.0	✓	Gaussian copula
Circles ($N_{\text{train}} = 400$)	0.0	0.0	✓	Gaussian copula
Moons ($N_{\text{train}} = 200$)	383666.5	0.0	✓	Centralized classifier
Moons ($N_{\text{train}} = 400$)	1097058.5	0.0	✓	Gaussian copula

Table D.5 Wilcoxon test for Gaussian copulas-based model and centralized classifier. Classifiers are logistic regressions.

Dataset	stat	p-value	CCl	Best method
Blobs ($N_{\text{train}} = 200$)	3596	0.0	✓	Gaussian copula
Blobs ($N_{\text{train}} = 400$)	0.0	0.0	✓	Gaussian copula
Circles ($N_{\text{train}} = 200$)	3068	0.0	✓	Gaussian copula
Circles ($N_{\text{train}} = 400$)	0.0	0.0	✓	Gaussian copula
Moons ($N_{\text{train}} = 200$)	1365479	0.0	✓	Gaussian copula
Moons ($N_{\text{train}} = 400$)	42095	0.0	✓	Gaussian copula

Table D.6 Wilcoxon test for Gaussian copulas-based model and maximally accurate individual classifier. Classifiers are logistic regressions.

Dataset	stat	p-value	CCl	Best method
Blobs ($N_{\text{train}} = 200$)	3596	0.0	✓	Optimal
Blobs ($N_{\text{train}} = 400$)	0.0	0.0	✓	Optimal
Circles ($N_{\text{train}} = 200$)	3068	0.0	✓	Optimal
Circles ($N_{\text{train}} = 400$)	0.0	0.0	✓	Optimal
Moons ($N_{\text{train}} = 200$)	1365479	0.0	✓	Optimal
Moons ($N_{\text{train}} = 400$)	42095	0.0	✓	Optimal

Table D.7 Wilcoxon test for Gaussian copulas-based model and optimal classifier. Classifiers are logistic regressions.

D.2 Wilcoxon results for real dataset

Dataset	stat	p-value	CCl	Best method
MNIST ($K = 2$)	16452	0.0	✓	Gaussian copula
MNIST ($K = 10$)	35381.5	0.0	✓	Gaussian copula
MNIST ($K = 50$)	12636	0.0	✓	Gaussian copula
20newsgroup ($K = 2$)	27189	0.0	✓	Independent copula
20newsgroup ($K = 10$)	9999.5	0.0	✓	Independent copula
20newsgroup ($K = 50$)	17341.5	0.0	✓	Independent copula
Satellite ($K = 2$)	25353	0.0	✓	Independent copula
Satellite ($K = 10$)	22866.5	0.0	✓	Independent copula
Satellite ($K = 50$)	54056.5	0.03693	✓	Independent copula

Table D.8 Wilcoxon test for Gaussian and independent copulas-based models. Classifiers are logistic regressions.

Dataset	stat	p-value	CCl	Best method
MNIST ($K = 2$)	0.0	0.0	✓	Gaussian copula
MNIST ($K = 10$)	0.0	0.0	✓	Gaussian copula
MNIST ($K = 50$)	0.0	0.0	✓	Gaussian copula
20newsgroup ($K = 2$)	0.0	0.0	✓	Gaussian copula
20newsgroup ($K = 10$)	0.0	0.0	✓	Gaussian copula
20newsgroup ($K = 50$)	0.0	0.0	✓	Gaussian copula
Satellite ($K = 2$)	0.0	0.0	✓	Gaussian copula
Satellite ($K = 10$)	0.0	0.0	✓	Gaussian copula
Satellite ($K = 50$)	0.0	0.0	✓	Gaussian copula

Table D.9 Wilcoxon test for Gaussian copula-based model and stacking. Classifiers are logistic regressions.

Dataset	stat	p-value	CCl	Best method
MNIST ($K = 2$)	1.0	0.0	✓	Gaussian copula
MNIST ($K = 10$)	0.0	0.0	✓	Gaussian copula
MNIST ($K = 50$)	0.0	0.0	✓	Gaussian copula
20newsgroup ($K = 2$)	57514.5	0.151	✗	Undecided
20newsgroup ($K = 10$)	0.0	0.0	✓	Gaussian copula
20newsgroup ($K = 50$)	0.0	0.0	✓	Gaussian copula
Satellite ($K = 2$)	44480	0.0	✓	Gaussian copula
Satellite ($K = 10$)	0.0	0.0	✓	Gaussian copula
Satellite ($K = 50$)	0.0	0.0	✓	Gaussian copula

Table D.10 Wilcoxon test for Gaussian copula-based model and classifier selection. Classifiers are logistic regressions.

Dataset	stat	p-value	CCl	Best method
MNIST ($K = 2$)	1.0	0.0	✓	Gaussian copula
MNIST ($K = 10$)	0.0	0.0	✓	Gaussian copula
MNIST ($K = 50$)	0.0	0.0	✓	Gaussian copula
20newsgroup ($K = 2$)	57514.5	0.151	✗	Undecided
20newsgroup ($K = 10$)	27.5	0.0	✓	Gaussian copula
20newsgroup ($K = 50$)	0.0	0.0	✓	Gaussian copula
Satellite ($K = 2$)	44661.	0.0	✓	Gaussian copula
Satellite ($K = 10$)	3461.5	0.0	✓	Gaussian copula
Satellite ($K = 50$)	11846.5	0.0	✓	Gaussian copula

Table D.11 Wilcoxon test for Gaussian copula-based model and weighted vote. Classifiers are logistic regressions.

Dataset	stat	p-value	CCl	Best method
MNIST ($K = 2$)	0.0	0.0	✓	Centralized classifier
MNIST ($K = 10$)	0.0	0.0	✓	Centralized classifier
MNIST ($K = 50$)	0.0	0.0	✓	Centralized classifier
20newsgroup ($K = 2$)	0.0	0.0	✓	Centralized classifier
20newsgroup ($K = 10$)	0.0	0.0	✓	Centralized classifier
20newsgroup ($K = 50$)	0.0	0.0	✓	Centralized classifier
Satellite ($K = 2$)	0.0	0.0	✓	Centralized classifier
Satellite ($K = 10$)	42506.5	0.0	✓	Centralized classifier
Satellite ($K = 50$)	0.0	0.0	✓	Centralized classifier

Table D.12 Wilcoxon test for Gaussian copula-based model and the centralized classifier. Classifiers are logistic regressions.

Dataset	stat	p-value	CCl	Best method
MNIST ($K = 2$)	2420.0	0.0	✓	Max. Ind. Clf.
MNIST ($K = 10$)	0.0	0.0	✓	Gaussian copula
MNIST ($K = 50$)	0.0	0.0	✓	Gaussian copula
20newsgroup ($K = 2$)	11851	0.0	✓	Max. Ind. Clf.
20newsgroup ($K = 10$)	0.0	0.0	✓	Gaussian copula
20newsgroup ($K = 50$)	0.0	0.0	✓	Gaussian copula
Satellite ($K = 2$)	30188.5	0.0	✓	Max. Ind. Clf.
Satellite ($K = 10$)	0.0	0.0	✓	Gaussian copula
Satellite ($K = 50$)	0.0	0.0	✓	Gaussian copula

Table D.13 Wilcoxon test for Gaussian copula-based model and the maximally accurate individual classifier. Classifiers are logistic regressions.

Appendix E

Detailed Wilcoxon signed-rank test results for copula and *t*-norm comparison

E.1 Wilcoxon test results related to section 3.5.6

Dataset	stat	*p*-value	CCl	Best method
Digits	1.0	0.0	✓	*t*-norm
Waveform	1122.0	0.0	✓	*t*-norm
Cancer	115.0	0.0	✓	*t*-norm
CNAE	0.0	0.0	✓	Gauss. copula
Segments	371.0	0.0	✓	*t*-norm
Wine	109.0	0.0	✓	*t*-norm
Mnist	0.0	0.0	✓	*t*-norm

Table E.1 Wilcoxon test for Gaussian copula-based model and the *t*-norm approach. Classifiers are decision trees.

Dataset	stat	p-value	CCl	Best method
Digits	36.0	0.0	✓	t-norm
Waveform	759.5	0.0	✓	t-norm
Cancer	1.0	0.0	✓	t-norm
CNAE	0.0	0.0	✓	t-norm
Segments	923.0	0.0	✓	t-norm
Wine	510.0	0.0	✓	t-norm
Mnist	0.0	0.0	✓	t-norm

Table E.2 Wilcoxon test for Gaussian copula-based model and the t-norm approach. Classifiers are logistic regressions.

Dataset	stat	p-value	CCl	Best method
Digits	0.0	0.034	✓	t-norm
Waveform	752.5	0.0	✓	t-norm
Cancer	0.0	0.0	✓	t-norm
CNAE	0.0	0.0	✓	t-norm
Segments	242.0	0.0	✓	t-norm
Wine	297.0	0.0	✓	t-norm
Mnist	0.0	0.0	✓	t-norm

Table E.3 Wilcoxon test for Gaussian copula-based model and the t-norm approach. Classifiers are Gaussian naive Bayesian .

E.2 Wilcoxon test results related to section 3.5.6

Dataset	stat	p-value	CCl	Best method
MNIST (K=2)	46633.0	0.0	✓	Gauss. copula
MNIST (K=10)	17879.0	0.0	✓	Gauss. copula
MNIST (K=50)	6574.5	0.0	✓	Gauss. copula
20newsgroup (K=2)	2703.5	0.0	✓	Gauss. copula
20newsgroup (K=10)	0.0	0.0	✓	Gauss. copula
20newsgroup (K=50)	0.0	0.0	✓	Gauss. copula
Satellite (K=2)	33093.5	0.0	✓	Gauss. copula
Satellite (K=10)	0.0	0.0	✓	Gauss. copula
Satellite (K=50)	304.5	0.0	✓	Gauss. copula

Table E.4 Wilcoxon test for Gaussian copula-based model and the t-norm approach. Classifiers are logistic regressions .

Dataset	stat	p-value	CCl	Best method
Blobs $N_{\text{train}} = 200$	1352927.5	0.0	✓	Gaussian copula
Blobs $N_{\text{train}} = 400$	613274.0	0.0	✓	Gaussian copula
Circles $N_{\text{train}} = 200$	1206413.0	0.0	✓	Gaussian copula
Circles $N_{\text{train}} = 400$	961941.5	0.0	✓	Gaussian copula
Moons $N_{\text{train}} = 200$	1518125.0	0.0	✓	t-norm
Moons $N_{\text{train}} = 400$	938542.0	0.0	✓	Gaussian copula

Table E.5 Wilcoxon test for Gaussian copulas-based model and t-norm approach. Classifiers are logistic regressions.