Sesmero, M.P., Ledezma, A.I. and Sanchis, A. (2015), Generating ensembles of heterogeneous classifiers using Stacked Generalization. WIREs Data Mining Knowl Discov, 5: 21-34.

DOI: 10.1002/widm.1143

Generating ensembles of heterogeneous classifiers using Stacked Generalization

M. Paz Sesmero•, Agapito I. Ledezma* and Araceli Sanchis

Over the last two decades, the machine learning and related communities have conducted numerous studies to improve the performance of a single classifier by combining several classifiers generated from one or more learning algorithms. Bagging and Boosting are the most representative examples of algorithms for generating homogeneous ensembles of classifiers. However, *Stacking* has become a commonly used technique for generating ensembles of heterogeneous classifiers since Wolpert presented his study entitled *Stacked Generalization* in 1992. Studies that have addressed the *Stacking* issue demonstrated that when selecting base learning algorithms for generating classifiers that are members of the ensemble, their learning parameters and the learning algorithm for generating the meta-classifier were critical issues. Most studies on this topic manually select the appropriate combination of base learning algorithms and their learning parameters. However, some other methods use automatic methods to determine good *Stacking* configurations instead of starting from these strong initial assumptions. In this paper, we describe Stacking and its variants and present several examples of application domains.

How to cite this article:
WIREs Data Mining Knowl Discov 2015. doi: 10.1002/widm.1143

INTRODUCTION

A classifier is a system that takes instances from a dataset and assigns a class or category to each of them. To perform this task, the classifier must have some type of knowledge. The classifiers can be created by using various forms of learning (e.g., deduction, analogy, or memorization), but the most common way of acquiring this knowledge is to infer it from a set of previously classified instances. This form of learning is called *supervised learning*.

Most research in *machine learning* has been devoted to developing methods that automate the classification tasks. Despite the variety and number of models that have been proposed, including artificial neural networks,[1] decision trees,[2] inductive logic programming,[3] and Bayesian learning algorithms,[4] the construction of a perfect classifier for any given task remains unobtainable.[5] Furthermore, no single approach can claim to be superior to any other.[6] Thus, the combination of different classification models is considered a viable alternative for obtaining more accurate classification systems. The strategy in ensemble systems is to create a set of classifiers and combine their outputs such that the combination outperforms all of the single classifiers. To achieve this goal, it is necessary to guarantee that (1) the individual classifiers are both accurate and diverse and (2) the output combination amplifies the correct decisions and cancels out the incorrect decisions.[7]

Studies in the ensemble field have typically focused on generating the ensemble members by applying a single learning algorithm and combining their outputs using a mathematical function. In contrast, *Stacking* generates the members of the *Stacking* ensemble using several learning algorithms and subsequently uses another algorithm to learn how to combine their outputs.

*Correspondence to: ledezma@inf.uc3m.es

Computer Science Department, Universidad Carlos III de Madrid, Madrid, Spain

Conflict of interest: The authors have declared no conflicts of interest for this article•.

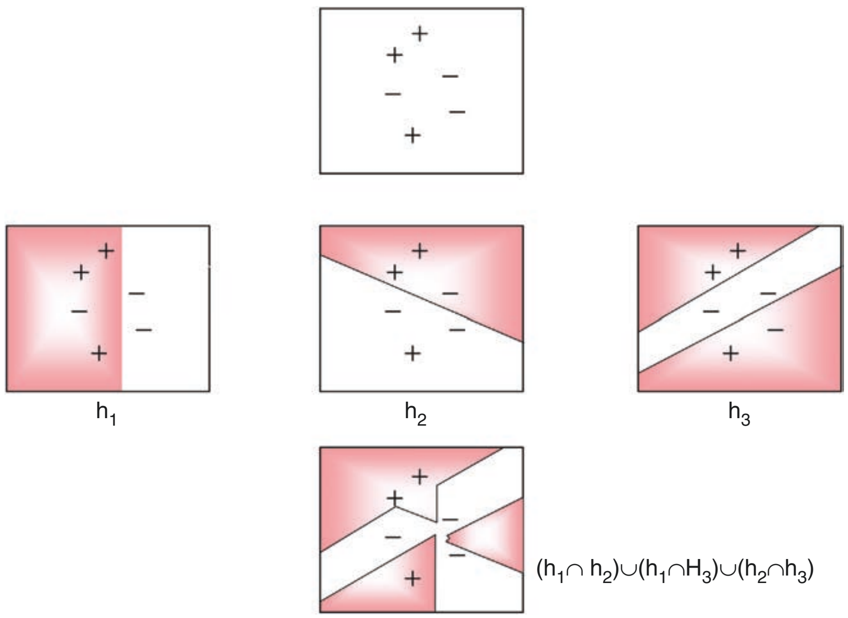

FIGURE 1 | Influence of diversity on the ensemble decision.

The remainder of this paper is organized as follows. First, some background on ensemble classifiers is given. Then, we present the main features of and a review of some of its variants, related approaches and recent applications. Finally, we draw some conclusions and discuss important topics about Stacking.

ENSEMBLES OF CLASSIFIERS

An ensemble of classifiers is a set of classifiers whose individual decisions are combined to obtain a system that hopefully outperforms all of its members.[8]

_ENREF_2Similar to what occurs with other systems in the field of artificial intelligence, the ensembles of classifiers respond to an attempt to emulate human behavior. Specifically, these systems try to replicate the performance of a human being when it faces an important decision. For example, it is common to ask the opinion of different doctors before having a surgery, performed or read reviews before buying a product. In other words, a decision is considered more reliable if it is made based on the opinion of different experts. Extrapolation of this proposition to the field of machine learning leads to the development of systems composed of several classifiers, in which the final decision is made collectively. This line of research in the machine learning field is known as the study of ensembles of classifiers.[9]

The strategy in ensemble systems is to create a set of accurate and diverse classifiers and combine their outputs such that the combination outperforms all the single classifiers. Therefore, classifier ensembles are built in two phases: generation and combination. In the generation phase, the individual components of the ensemble, known as base classifiers, are generated. In the combination phase, the decisions made by the members of the ensemble are combined to obtain one decision. A detailed description of these phases is provided in the following subsection.

Generating Base Classifiers

To obtain an ensemble of classifiers that outperforms all its members, the base learners must be both accurate and diverse. A classifier is accurate when its classification error is lower than that obtained when the classes are randomly assigned. Two classifiers are diverse if they make errors at different instances.

Demanding accurate classifiers appears to be a logical requirement; the combination of a set of incorrect decisions cannot easily generate a correct hypothesis. To illustrate why diversity is a necessary condition, consider, in a two classes domain, an ensemble of three classifiers, h_1, h_2, and h_3, and a new example x that must be classified. If the three classifiers are not diverse, then when the decision given by h_1 is wrong, the decisions given by h_2 and h_3 will also be wrong. Therefore, the final ensemble decision will be wrong. However, if the base classifiers are diverse, the decisions given by both h_2 and h_3 will be correct even when the decision given by h_1 is wrong. Therefore, the ensemble decision will be correct if

TABLE 1 | Summary of Diversity Measures

Name	Symbol	Definition	↑/↓[1]	Pairwise
Q statistic	Q	$\frac{N^{11}N^{00}-N^{01}N^{10}}{N^{11}N^{00}+N^{01}N^{10}}$	↓	Y
Correlation coefficient	ρ	$\frac{N^{11}N^{00}-N^{01}N^{10}}{\sqrt{(N^{11}+N^{10})(N^{01}+N^{00})(N^{11}+N^{01})(N^{10}+N^{00})}}$	↓	Y
Fail/non-fail disagreement measure	dis	$\frac{N^{01}+N^{10}}{N^{11}+N^{10}+N^{01}+N^{00}}$	↑	Y
Double-fault measure	DF	$\frac{N^{00}}{N^{11}+N^{10}+N^{01}+N^{00}}$	↓	Y
Kappa degree-of- agreement statistic	κ	$\frac{\frac{\sum_{i=1}^{K} N_{ii}}{N}-\sum_{i=1}^{K}\left(\frac{N_{i*}}{N}\frac{N_{*i}}{N}\right)}{1-\sum_{i=1}^{K}\left(\frac{N_{i*}}{N}\frac{N_{*i}}{N}\right)}$	↓	Y
Plain disagreement measure	Div_plain	$\frac{1}{N}\sum_{k=1}^{N} Is\left(C_i\left(x_k\right), = C_j\left(x_k\right)\right)$	↑	Y
Ambiguity	Amb	$\frac{1}{LKN}\sum_{l=1}^{L}\sum_{n=1}^{N}\sum_{k=1}^{K}\left(Is\left(C_l\left(x_k\right)=k\right)-\frac{N_k^n}{L}\right)$	↑	N
Entropy	E	$\frac{1}{N}\sum_{n=1}^{N}\frac{1}{\left(L-\frac{L}{2}\right)}\min\left(l\left(x_k\right), L-l\left(x_k\right)\right)$	↑	N

N, cardinality of the test set; K, number of classes; L, the number of base classifiers; N^{ab} is the number of instances in the dataset, classified correctly (a = 1) or incorrectly (a = 0) by the classifier i, and correctly (b = 1) or incorrectly (b = 0) by the classifier j; N_{ij}, number of instances in the dataset, labeled as class i by the first classifier and as class j by the second classifier; $C_i(x_k)$, class assigned by classifier i to instance k; $Is(c)$, a Boolean function. Its value is 1 if c is true and 0 if c is false; N_k^n, number of base classifiers that assign instance n to class k; and $l(x_k)$, number of classifiers that correctly classified instance k.
[1]Monotonically increasing/decreasing measures are identified with an ascending/descending arrow, respectively.

all the decisions have the same relevance. Figure 1 illustrates this example graphically.

Diversity is a necessary condition for obtaining a good ensemble. However, measuring diversity is not straightforward because there is no formal definition of diversity and no consensus on how to quantify this magnitude.[10] Some of the more common ways to quantify ensemble diversity are shown in Table 1.

We have analyzed the relevancy of diversity among the base classifiers and how to quantify it. It is now necessary to review the most well-known techniques for generating diverse classifiers.

The techniques used to generate diverse classifiers are based on the idea that the hypothesis of a classifier depends on both the learning algorithm and the subset used to generate these classifiers. Therefore, it is possible to generate classifiers whose decisions are dissimilar from each other by varying the training set and/or learning algorithm.

Three different approaches can be used to generate an ensemble of classifiers[9]:

- *Resampling the training examples:* This approach includes two of the most widely known methods for constructing classifier ensembles: *Bagging*[11] and *Boosting*.[12] *Bagging* builds different versions of the training set by sampling with replacement. In contrast, •*Boosting* obtains the different training sets by focusing on the instances that are misclassified by the previously trained classifiers.
- *Manipulating the input features*. Another way to achieve diversity between classifiers is by modifying the set of attributes used to describe the instances.[13–17]
- *Manipulating the output target*: Another approach for generating a pool of diverse classifiers is having each classifier solve a different classification problem. This category includes methods that solve multiclass problems by converting them into several binary subproblems. Among the strategies for decomposing a multi-class problem into two-class problems are one-against-one (OAO),[18] one-against-all (OAA),[19] one-against-higher-order (OAHO)[20] and error correcting output codes (ECOC).[21] Other systems that decompose the multiclass problem into several pairwise subproblems, such as binary-complementary-ensemble (BCE)[22,23] and complementary-complementary ensemble (CCE)[24] can be grouped into this approach.

Methods that vary the learning algorithm can be subdivided in two groups:

- Approaches that use *different versions of the same learning algorithm*. Kolen and Pollak[25] demonstrated that a pool of artificial neural networks starting from different initial weights can be trained to generate diverse classifiers and thus a good ensemble. Alternatively, a pool of diverse decision trees can be obtained by varying the criterion used to expand a C4.5. node.[26]
- Approaches where diversity is obtained using *different learning algorithms*. According to Wolpert,[27] ensembles with base classifiers trained from different learning algorithms (heterogeneous ensembles) exploit the different biases of each learning algorithm. Therefore, most studies in the field of ensembles have focused on the combination of different inducers, such as artificial neural networks, decision trees, Bayesian models, nearest neighbor, and support vector machines. As will be shown below, Stacking[27] and most of its variants achieve diversity by applying this approach.

Integrating Decisions

Once the base classifiers that comprise the ensemble have been built, the next step is to establish a procedure through which the individual decisions are combined to obtain a final hypothesis. There are two main strategies for combining classifiers: fusion and selection.[6,28] Classifier selection presupposes that each classifier is an expert in some local region of the space. Therefore, when an instance is submitted for classification, the ensemble decision coincides with the decision given by the classifier responsible for the region of the space to which the instance belongs.[29] In classifier fusion, the decisions from all members of the ensemble are combined in some manner to make the ensemble decision. Classifier fusion algorithms include combining rules, such as the average, majority vote, weighted majority vote, and the Borda Count, and more complex integration models, such as meta-classifiers. A meta-classifier is a second-level classifier generated from the outputs given by the base learners. According to Rokach,[30] Stacking, arbiter tree,[31] combiner tree[32] and the Grading approaches[33] are considered integration methods based on meta-learning.

STACKED GENERALIZATION

Stacking is short for Stacked Generalization.[27] As noted above, unlike other ensemble generation algorithms, such as *Bagging* or *Boosting*, which generate an ensemble of classifiers using the same learning algorithm (*homogeneous ensembles*), *Stacking* generates an ensemble composed of *heterogeneous classifiers*. Because each learning algorithm uses different methods to represent the knowledge and different learning biases, the hypothesis space will be explored from different perspectives with the aim of generating a pool of diverse classifiers. Therefore, when their predictions are combined, the resultant model is expected to be more accurate than each individual member.

To combine the individual predictions of the ensemble members, *Stacking* uses the concept of *meta-classifiers* or *meta-learners*. The *meta-classifier* or level-1 model is generated using a learning algorithm following a cross-validation-like process. This classifier attempts to model how the outputs of the base classifiers or level-0 models should be combined to generate the final output. Figure 2 provides a general overview of the *Stacking* process.

Stacking is an ensemble of classifiers in which (1) the base learners are trained using different training parameters (generally different learning algorithms) and (2) the outputs of the base learners are combined by using a meta-classifier. One of the issues in *Stacking* is obtaining the appropriate combination of base-level classifiers and the meta-classifier, especially in relation to each specific dataset. If only a small number of classifiers and algorithms will be used, this problem can be solved by a simple method, namely, exhaustive search, in a reasonable amount of time. However, it is difficult to determine the best *Stacking* configuration when the search space is large.

Formal Definition

Given a dataset S, *Stacking* first generates randomly a subset of equal size datasets $S_1, \ldots, S_T$ and subsequently follows a process similar to a J-fold cross-validation process: it omits one of the subsets (e.g., S_j) to be used later. The remaining instances $S^{(-j)} = S - S_j$ are used to generate the level-0 classifiers by applying K learning algorithms, $k = 1, \ldots, K$, to obtain K classifiers. $S^{(-j)}$ and S_j are the training and test sets respectively of the j-th fold in the cross-validation. After the level-0 models have been generated, the S_j set will be used to generate the *level-1* instances. Level-1 training data are generated from the predictions of the level-0 models over the instances in S_j, which were omitted for this purpose (Figure 3a). Level-1 data have K attributes, whose values are the predictions of each one of the K level-0 classifiers for every instance in S_j. At the end of the cross-validation process, each level-1 training example will be composed for K attributes (the K predictions) and the target class, which is the real class value for every instance in S. Once the level-1 data have been built from all instances in S, any learning algorithm can be used to generate the level-1 model

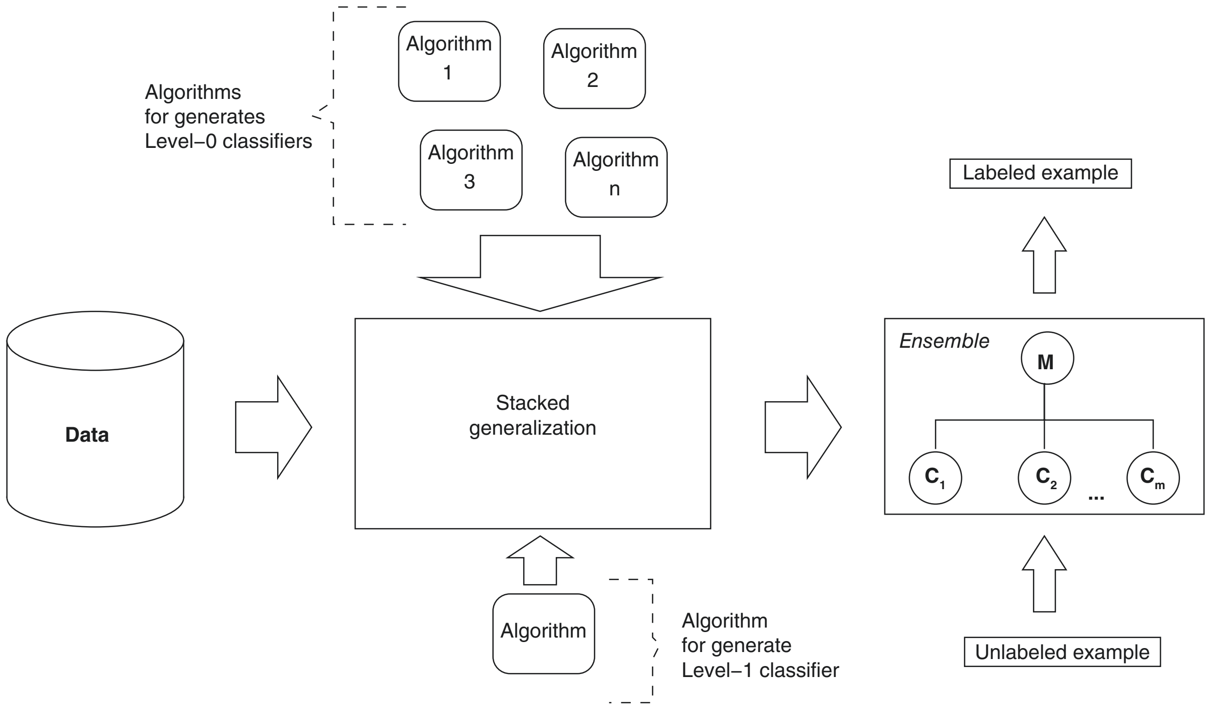

FIGURE 2 | Overview of the stacking procedure.

(Figure 3b). To complete the process, the level-0 models are re-generated from the entire dataset *S* (it is expected that this process improves the accuracy of the classifiers slightly) (Figure 3c). In Figure 3d the final ensemble structure generated by Stacking is shown. To classify a new instance, the level-0 models produce a vector of predictions that is the input to the level-1 model, which in turn predicts the class.

STACKING VARIANTS AND RELATED APPROACHES

Since Wolpert first proposed *Stacking* in 1992, several related studies have been published. In general terms, these studies can be grouped into two categories: those that address the *Stacking* parameter selection and those that present approaches similar to *Stacking*. We provide a brief review of these two types of studies in the following subsections.

Stacking Variants

As initially noted by Wolpert,[27] some issues of *Stacking* are considered black art, such as the selection of base classifiers, the type of meta-data and the classifier to be used in level-1. Some studies that address these issues and other related topics are presented below.

Skalak[34] proposed the use of instance-based learning classifiers that store a few prototypes per class as level-0 classifiers. They also proposed to use a decision tree as a meta-classifier or level-1 classifier. Fan et al.[35] proposed to determine the overall accuracy of the ensemble generated by Stacking using *conflict-based accuracy estimates*. The authors use two tree-based classifiers and one rule-based classifier as base-level classifiers. In contrast, for the meta-level, they use a rote table that behaves as a decision tree without pruning in this case. This *Stacking* configuration is evaluated using four datasets (including two artificial datasets). Although the authors claim that the proposed measure is superior to all existing measures, their results do not clearly demonstrate that this estimate can be generalized to more datasets or other meta-classifiers.

Merz[36] proposed a variant of *Stacking* that uses correspondence analysis to detect correlations between base-level classifiers. Once dependencies have been removed from the original meta-level space, a nearest neighbor method (meta-level algorithm) is applied over the resulting feature space. This approach is called *SCANN*.

Ting and Witten[37] address two Stacking configuration issues: level-1 classifier types and the data types of the meta-level. They propose the use of class probabilities rather than a single class prediction as

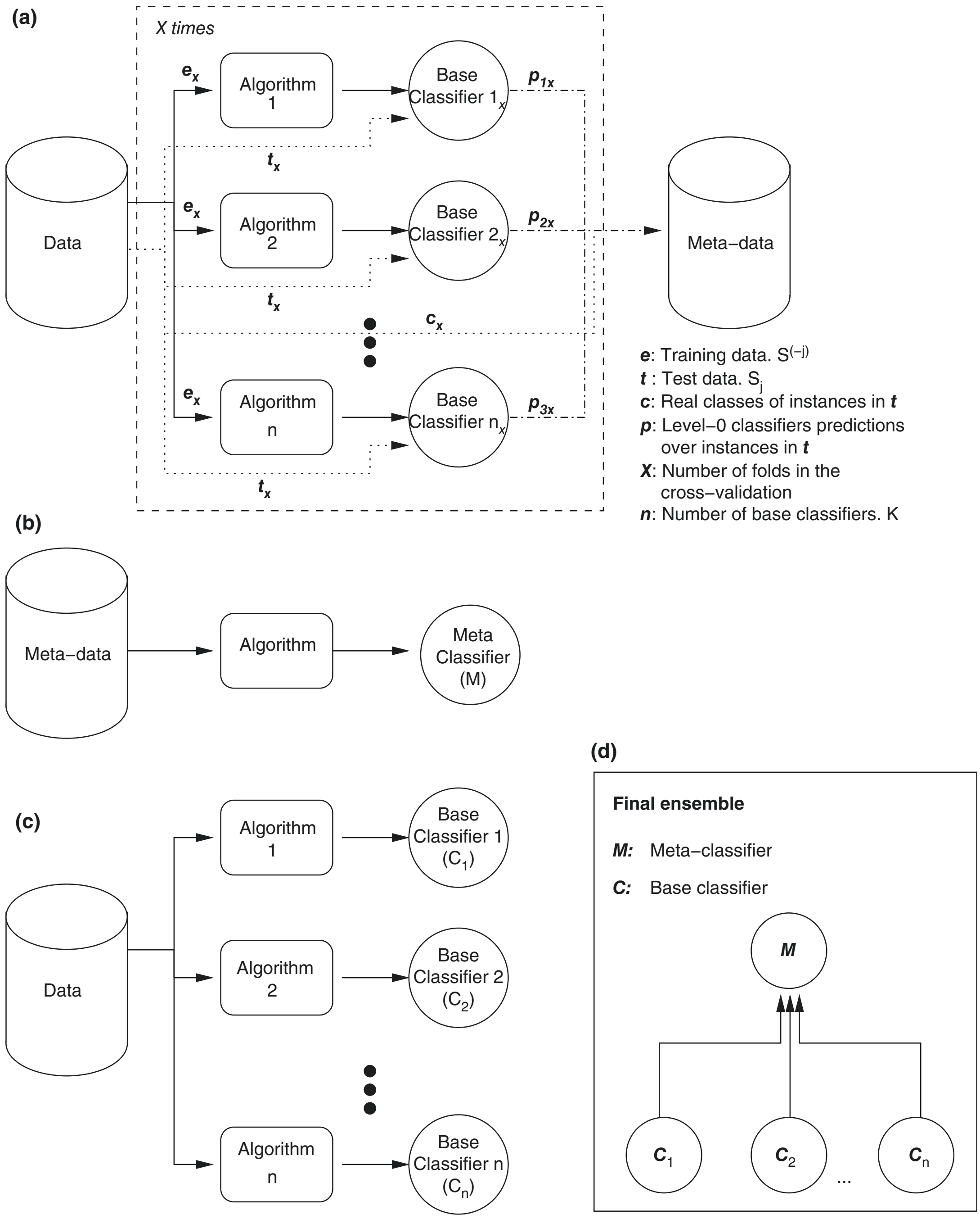

FIGURE 3 | Generating an ensemble of classifiers using Stacking.

outputs of the level-0 classifiers. Thus, each instance of the meta-level is composed of the class probabilities given for each level-0 classifier, followed by the actual class of the instance. The authors argue that by using the class probabilities as meta-data, Stacking uses both the prediction and confidence of the base-level classifiers. Regarding the type of meta-level classifier to be used, the authors conclude that multi-response linear regression (MLR) is the most appropriate algorithm for generating the meta-level model, at least when using class probabilities as meta-level data. Moreover, Ting and Witten studied the necessity of non-negative constraints for the attribute weights in linear models because both Breiman[38] and LeBlanc and Tibshirani[39] report the need to use nonnegative constraints when using *Stacking* in a regression task. They concluded that non-negative restrictions are not necessary in *Stacking* to improve the overall accuracy of the ensemble when performing a classification task. However, these restrictions are useful for improving the interpretability of the level-1 model.

Based on work of Ting and Witten,[37] Seewald[40] used MLR as the level-1 classifier but with a different set of attributes in the meta-level to overcome a weakness of *Stacking* with MLR (SMLR) in domains with more than two classes. This weakness was not present in the original version of *Stacking*, and Seewald argues that this new weakness may be due to the dimensionality of the meta-data. When MLR is used as the meta-classifier, a linear equation for each class is constructed using the class probability distributions given by the base classifiers. *StackingC*, as this variant is known, is proposed to use only the class probabilities associated with the class to which the linear model is built, thus reducing the dimensionality of the attributes of the meta-level by a factor equal to the number of classes. The results of this research indicate an improvement over SMLR when used with the full set of probability distributions. Furthermore, Seewald argues that the observed improvement is due to not only the reduction of the dimensionality of the meta-data but also the high diversity of class models generated at the meta-level.

Todorovski and Džeroski[41] proposed a variant of *Stacking* that uses a decision tree approach as the learning method in the meta-level. This method, called *meta decision trees* (MDTs), replaces class-value predictions in its leaf nodes by the name of the base-level classifier that should be used to obtain the class for a specific example. The meta-level data are composed of properties of the probability distributions that reflect the confidence of the base-level classifiers (e.g., entropy and maximum probability) rather than the distributions themselves. These properties are used to generate small MDTs.

Džeroski and Ženko[42] proposed two additional variants of *Stacking*. The first variant addresses the issue of the type of meta-data based on SMLR proposed by Ting and Witten.[37] The authors propose an extension of meta-data, adding two additional sets of attributes: the probability distributions multiplied by the maximum probability and the entropies of the probability distributions. Moreover, Džeroski and Ženko proposed another extension of SMLR in which they replace the linear regression approach by a tree induction approach as the meta-level model. They called this method *Stacking with multi-response model trees* (SMRMT). According to the authors, comparing different *Stacking* approaches, SCANN, SMDTs, SMLR, and the SelectBest scheme (selecting the best classifier with cross-validation) appear to perform at approximately the same level. Moreover, Džeroski and Ženko concluded that SMRMT outperforms previous *Stacking* variants, including *StackingC*, and selects the best classifier from the ensemble by cross-validation.

Menahem et al.[43] proposed a new variant of *Stacking* called *Troika*, whose main feature is that the meta-level is composed of three layers. In the first layer, the outputs of the base classifiers are combined using a *OAO* ensemble, whose members are called specialist classifiers. The goal of each specialist is to predict the probability that an instance belongs to one of the two classes that it distinguishes. In the second stage, the outputs of the specialists are combined again using a *OAA* schema. The task of the level-2 classifiers is to learn the behavior patterns of the specialist classifiers and to predict whether the output given by a specialist is correct. The third layer contains a classifier and produces the ensemble final decision. Moreover, the authors analyze three arrangements to train the base classifiers (OAO, OAA, and all-against-all) and determine that *Troika* is more accurate than *Stacking* and *StackingC* in all cases. Regarding the runtime, the authors conclude than *Troika* outperforms *Stacking* and *StackingC* only when the base classifiers are trained using the OAO architecture.

Ledezma et al.[44] proposed an approach to determine good *Stacking* configurations by a genetic search. Their approach, called *GA-Stacking*, not only determines which meta-level and which (and how many) base classifiers must be present but also their learning parameters. Moreover, *GA-Stacking* provides flexibility and extensibility compared to previous *Stacking* variants because it can easily incorporate new learning algorithms and is not restricted by ‘a priori’ assumptions. Moreover, *GA-Stacking* adapts the *Stacking* configuration to the domain biases and characteristics so that the *Stacking* configurations determined by *GA-Stacking* are domain dependent. However, GA-Stacking requires a longer execution time than the other approaches to obtain a specific *Stacking* configuration.

Following a similar approach to the work of Ledezma et al.[44] and posing the *Stacking* configuration as an optimization problem, Chen and Wong[45] proposed the use of ant colony optimization (ACO) to determine domain-dependent *Stacking* configurations. They use the meta-heuristic ACO to determine the level-0 Stacking classifiers with a predefined level-1 classifier[45] as well as the entire Stacking system configuration (level-0 and level-1).[46]

Recently, Shunmugapriya and Kanmani[47] proposed the use of another meta-heuristic search algorithm to determine which and how many base classifiers to use and what meta-classifier to use based on the domain. Therefore, they have proposed to use an artificial bee colony (ABC) method. The authors compared their results with the studies of Ledezma et al.[44] and Chen and Wong,[45] and they conclude that the results of the *Stacking* configurations determined by ABC are comparable to those obtained in the previous study.

The approaches introduced above are compared in Table 2 with regard to the focus area, number of base classifiers, algorithms for base-classifier generation, type of meta-data, and meta-level classifier. In addition, some observations are included.

Related Approaches

In addition to the *Stacking* variants already discussed, there are studies that can be viewed as either *Stacking*-based implementations or studies that hold many similarities with *Stacking*.

Chan and Stolfo[48] proposed a strategy similar to *Stacking*, which they called *Combiner*. The main idea behind the *Combiner*, as the authors claim, is to merge the predictions of base classifiers by learning the relation between the predictions of base classifiers and the correct prediction. Moreover, they propose a variant of the combiner strategy called *attributes combiner*, in which the attributes of the meta-level are composed of not only the predictions of a class but also the original attributes of the instance. As shown in Schaffer's study of Stacking bi-level,[49] this approach can reduce the performance of the ensemble. In contrast, Chan and Stolfo[48] proposed an approach that uses what they call an *Arbiter*, which is a classifier independent from the remaining base classifiers that is trained on a subset of the original dataset. This subset of the data consists of the instances in which the base classifiers present diverse predictions. The purpose of an arbiter is to provide an alternative and more elaborate prediction when base classifiers present contradictions. In addition, Chan and Stolfo proposed what they call an *arbiter tree*, in which arbiters that specialize in resolving conflicts between pairs of classifiers are arranged in a binary decision tree. To carry out the classification of an instance, the method starts from the leaf nodes formed by base classifiers and goes up through the tree to the root node that provides the final classification.

Ting[50] proposed a composite learner framework that selects the classification that is estimated to have the higher accuracy as the final prediction of the ensemble. This framework uses the predictions of the base classifiers to learn a function that reflects the inner measure of confidence of the algorithm on an estimate of their accuracy on the output. This function can be used to combine the expertise of the classifier.

Gama and Brazdil[51] proposed a method closely related to *Stacking* that they called *Cascade Generalization*. In this method, the classifiers are applied sequentially, and there is no meta-classifier. When each base classifier is applied to the data, it increases the number of attributes of the dataset by adding the class probability distribution. The following classifier then uses this new dataset so that the order in which classifiers are used becomes an important factor.

Seewald and Fürnkranz[33] proposed a scheme known as *Grading*. This scheme creates a meta-level classifier for each level-0 classifier. The learning task for each level-1 classifier is to predict whether the level-0 classifier prediction will be correct. The meta-level data are composed of base-level attributes, and the class values are *correct* or *incorrect*. The final prediction of the ensemble is calculated through a weighted voting mechanism over the predictions of the base classifiers. The weight assigned to the vote of each base classifier is the confidence that his prediction will be correct. This weight is estimated by the meta-classifier associated with the base classifier. This work has some similarities with the work performed by Ting.[50]

Torres-Sospedra et al.[52] proposed a combination strategy based on ANN in which the predictions of the level-0 classifiers for the entire training set are used to train the meta-classifier. Based on this idea, they propose two different combination schemes: *Stacked* and *Stacked+*. In both schemes, the outputs provided by the base learners are used as inputs to the meta-level, but in *Stacked+*, the original input data are also used as inputs to the meta-classifier.

Inspired by the work of Wolpert[27], Cohen and Carvalho[53] proposed a stacked of classifiers to be applied in *sequential partitioning tasks*. This meta-learning method, called *stacked sequential learning*, *SSL*, that seeks to augment an arbitrary base learner in sequential learning problems.[54] In this approach, during the training phase, a cross-validation process is carried out in order to obtain the predicted labels, which are joined with the original input features vector, taking into account a neighborhood around the examples. With this training dataset—that they called extended dataset—a *metalearner* is built and a base learner is obtained from the original dataset. Then, in the inference phase, when a new instance arrives, the base learner is used to generate the prediction label for the instance. After

TABLE 2 | A comparison of Focus Area, Base, and Meta-Level Parameters of Stacking-Based Approaches

Year	Authors	Focus Area	Number of Base Classifiers	Algorithms for Base Classifiers	Type of Metadata	Meta Classifier	Observations
1997	Skalak[34]	Base classifier selection	2, 3 and more than 3 (3, 5, 11, 21)	Instance-based classifiers (simple nearest neighbor classifier)	Class predictions	Voting, Nearest Neighbor and ID3	Homogeneous ensemble
1999	Merz[36]	Meta-level	5 to 8	Back propagation neural network, CN2, C4.5, OC1, OC1 variant, PEBLS, 1-NN, naïve Bayes	Class predictions represented in a space of uncorrelated dimensions	Nearest neighbor	Correspondence analysis (errors)
1999	Fan et al.[35]	Overall accuracy of the ensemble	2 and 3	Ripper, Cart, ID3 and C4.5	Class predictions	A rote table (it functions as an un-pruned full decision tree)	New metrics: conflict-based accuracy estimate and conflict-based accuracy improvement estimate
1999	Ting and Witten[37]	Meta-level	3	C4.5, naïve Bayes and IB1	Class probability distributions	Multi-response linear regression (MLR)	Represents meta-level data as a class probability vector
2002	Seewald[40]	Meta-level data	6	Decision table, C4.5, naïve Bayes, kernel density, MLR and K*	Reduced class probability distributions	MLR	A Stacking variant called StackingC
2000	Todorovski and Džeroski[72]	Meta-level data and learner	5	C4.5, LTree, CN2, k-NN and naïve Bayes	Class probability distribution properties (e.g., entropy and maximum probability)	Meta decision trees (MDTs)	A Stacking variant called Stacking with MDTs
2004	Džeroski and Ženko[73]	Meta-level data and learner	3 and 7	C4.5, k-NN, naïve Bayes, K*, kernel density estimation, decision table, MLR	Class probability distributions and class probability distribution augmented with two additional calculated attributes	Multi-response model trees (MRMT) and MLR	Two Stacking variants
2009	Menahem et al.[43]	Meta-level classifier	1, 3 and 6	C4.5, VFI, IBk, PART, Bayes-Net, SMO.	Class probability distributions	Three stages in logistic algorithm	A Stacking variant in which the meta-level is split into three layers. Base classifiers are trained using two different binarization methods (OAA, OAO) and the AAA scheme.
2010	Ledezma et al.[55]	Whole Stacking system	Variable. up to 10	C4.5, naïve Bayes, simple naïve Bayes, IBk, PART, DT, decision stump, random forest, random tree, MLR, MRMT, K*, VFI, conjunctive rule, JRip, Nnge, hyper-pipes	Probability distributions	Variable. Selected by a genetic algorithm	Uses genetic algorithms for the parameter settings of Stacking: GA-Stacking
2011	Chen and Wong[45,46]	Entire Stacking system	Variable	Naïve Bayes, logistic classifier, IB1, IBk, K*, OneR, PART, ZeroR, decision stump, C4.5	Probability distributions	C4.5[45] and selected by the ant colony[46]	Uses an ant colony optimization technique for the parameter settings of Stacking: ACO-Stacking
2013	Shunmugapriya and Kanmani[47]	Entire Stacking system	Variable. up to 10	Naïve Bayes, logistic classifier, IB1, IBk, K*, OneR, PART, ZeroR, decision stump, C4.5	Probability distributions	Variable. Selected by the artificial bee colony algorithm	Uses a bee colony algorithm for the parameter settings of Stacking: ABC-Stacking

this label generation, the extended instance is created so that the metalearner can use it to produce the final prediction.

Based on *GA-Stacking*[44,55]_ENREF_55, Ordoñez et al.[56] proposed an approach that uses genetic algorithms to determine which base classifiers must be present in the ensemble as well as the method used to combine these classifiers. Although the final ensemble uses a meta-classifier as the decision combination method in some cases, in other cases, it uses other methods of combining base classifier decisions. Hence, it is considered a related work and not a *Stacking* variant.

Based on work of Cohen and Carvalho,[53] Gatta et al.[57] proposed a new framework whose goal is to capture the data interactions using a *neighborhood function*. In this way, the meta-classifier is trained using an extended training set that is composed by the original data set and the output given by this neighborhood function. To evaluate the performance of this general framework, called MS-SSL, two implementations of the neighborhood function were proposed. These implementations were based on the combination of two different Multi-Scale Decomposition schemes (a pyramidal decomposition and a multi-resolution decomposition) and a sampling pattern. Both models (*Pyr-SSL* and *MR-SSL*) were built using *AdaBoost*, with a maximum of 100 decision stumps, as classification algorithm. The proposed systems were evaluated in two domains: a text categorization task and an image pixel classification problem. According to the authors, experimental results proved that, in both domains, *MS-SSL* outperforms classical *SSL*, *CRF*[58] and *AdaBoost*. A drawback presents in *MS-SSL* is the impossibility of dealing with multiclass problems. One way to address this difficulty is modifying the neighborhood function and replacing the base classifiers used in the original *MS-SSL* scheme by others capable of dealing with data belonging to N classes. This adaptation, called *MMSSL*, is presented and applied for the resolution of several multi-class sequential learning problems in.[59] In this study, the authors concluded that MMSSL shows significant performance improvement compared with classical approaches. Moreover, authors note that MMSSL is able to keep the relationship among classes at different scales. Therefore, from a qualitative point of view, it is possible to state that the results of MMSSL are better than those obtained with the rest of the evaluated models.

Trivedi and Kapadia[60] suggested improving the ensemble accuracy by combining the philosophies of both *Stacking* and *Boosting*. The proposed algorithm, named '*sequential stacking*', trains the base classifier sequentially, giving more importance to instances that were misclassified by the previous classifiers. After the training, the outputs of the level-0 classifiers are used to train the meta-classifier. Therefore, the diversity among the base classifiers is achieved using a version of Boosting instead of cross-validation.

Applying Stacking

Most works related to Stacking have focused on determining an answer to what Wolpert called black art. However, there is a series of studies focused on the application of Stacking in real domains. We present some of the most representative examples below.

Doumpos and Zopounidis[61] used Stacking to distinguish potential defaulters from non-defaulters. Their work is focused on the combination of seven classification algorithms (linear discriminant functions, quadratic discriminant functions, logistic functions, probabilistic neural networks, nearest neighbors, decision trees, and support vector machines), which have been successfully used in previous studies on credit risk assessment. The outputs of the level-0 classifiers are transformed by applying a principal component analysis and are subsequently sent to the meta-classifier. To complete the study, seven different meta-classifiers were implemented (each using one of the seven above-mentioned classification algorithms) and 54 different scenarios (different combinations of the characteristic parameters of each classification algorithms) on three datasets were analyzed. According to the authors, although the experimental results are affected by the value of the classification algorithm parameters, the models based on Stacking are more efficient than the single-method models. Moreover, they observed that the exclusion of a level-0 classifier does not necessarily reduce the Stacking performance.

Hu and Tsoukas[62] applied a method based on the *Stacking* methodology to identify the factors that affect consumer choices. The main goal of this study was to investigate the role of demographic and situational factors on consumer choices. In their investigation, they use classifier ensembles composed only of ANNs. According to the authors, all implemented models benefited from stacked generalization, and the best models are those that contain exclusively situational variables.

Qian and Rasheed analyzed *Stacking* and *Voting* as tools to predict the trend of the Dow Jones index[63] and the trend of the exchange spot rate of the US dollar against the British pound trend.[64] In their investigation, they used artificial neural networks, decision trees and k-nearest neighbors as level-0 classifiers but provided no information about the meta-classifier. In both studies, they conclude that Stacking and Voting have a similar performance, and in both cases, their gain in relation to the best level-0 classifier is null. The authors argue that this result is due to the lack of diversity among the level-0 classifiers.

The application of the main idea behind the Stacked Generalization is known as *blending of classifiers* in some domains.[65] Such is the case of the work of Sill et.al[66] in which the authors present the application of Stacking as a key facet of the second place team solution to the Netflix Prize Competition.[67] In this work, the authors presented a meta-level linear technique, known as Feature-Weighted Linear Stacking. This technique combines the base classifiers predictions linearly through coefficients that are themselves linear functions of some additional inputs known as meta-features. This approach demonstrates an accuracy improvement over the standard linear stacking. However, as authors claim, the creation of useful meta-features is an art, so it depends on the application domain.

Escalera et al.,[68] applied Stacked Sequential Learning[53](*SSL*)to a problem related with laughter detection. In their study, they proposed the fusion of both audio and video cues to deal with the laughter recognition in face-to-face conversations. Once the audio and the visual cues have been identified, processed, and merged to obtain a unique feature vector, a level-0 classifier is trained. Then, according to *the* SSL scheme, an extended data set is created which joins the original training data features with the predicted labels produced by the level-0 classifier. Finally, this extended data set is used as input to a second classifier. The experimental results demonstrated that the proposed model outperforms *AdaBoost*. Nevertheless, the use of both audio and visual cues does not seem to improve the results obtained when only audio features are used.

A study developed using work from computer science, psychiatry, and nuclear medicine[69] analyzed the use of *Stacking* to differentiate Alzheimer's disease and mild cognitive impairment. Because medicine offers a host of tools designed to help the physician in his diagnostic process, in this study, Stacking is viewed as not only a method for building heterogonous ensembles but also a method for integrating the decisions from different diagnostic tools, including PET scans, Consortium to Establish a Registry of Alzheimer's Disease, mini-mental state examination and clock drawing tests. Therefore, each level-0 classifier is k-NN trained from data from one of the sources. The experimental results demonstrated that the mean accuracy of a simple k-NN including all of the features was 76%, whereas the mean accuracy of Stacking was 83%. Thus, Stacking achieved an accuracy gain of 7%.

StackTIS[70] is a *Stacking*-based methodology whose objective is the detection of potential translation initiation sites (TISs). The proposed model is based on the combination of three different classifiers, where each classifier learns from data described by different attributes. The first classifier is an SVM that is trained to identify the coding potential of a cDNA sequence. This classifier uses the 64 codon frequencies as input. The second classifier is a first-order homogeneous Markov chain, whose inputs are the segment of cDNA that enclose an ATG codon (starting from position −7 and ending at position +5). The third classifier is a heuristic model that calculates the probabilities of an ATG to be the TIS based on its distance from the 5'. Finally, the predictions given by these three components are used as input to the meta-classifier. In this work, two different learning algorithms were considered as meta-classifiers, namely, MLR and M5',[71] but the experimental results indicate than M5' outperforms MLR slightly. StackTIS was tested on two human datasets and one rice dataset. According to the authors, for the three evaluated domains, StackTIS outperforms other popular approaches that are common in the TIS prediction literature.

Razmara and Sarkar (•Razmara & Sarkar, 2013) applied an Stacking variant to the field of the Statistical Machine Translation. In their investigation, the authors adopted the approach suggested by Wolpert under which cross-validation can be used to construct different weak classifiers. So, Razmara and Sarkar propose building an homogeneous ensemble in which each base classifier–translation model implemented using a statistical machine translation system called *Kriya* (Sankaran, Razmara, & Sarkar, 2012) – is training using *k-1* partitions of the data. Then, the remaining data partition is used to tune the base learner parameters. The hypothesis from these base classifiers are combined in a second module called *Ensemble Decoding*. To provide a greater flexibility in its answer – scores–, the *Ensemble Decoding* module is prepared to handle different mixture operations: weighted sum, weighted max, model switching, and product. Experimental evaluation on two language pairs showed that the proposed model outperforms Bayesian Model Averaging and, in most cases the ensemble outperforms every one of its base translator.

AQ4

CONCLUSION

Today, it is common to use algorithms, such as *Bagging* and *Boosting*, to generate ensembles of classifiers as a standard method in classification tasks. Such techniques are implemented in a large number of data mining tools, which facilitates their use and evaluation. Thus, many studies have focused on the application of these techniques in a variety of domains. However, after more than two decades since the publication of Wolpert's paper, the use of Stacking in real applications remains relatively rare, possibly due to what Wolpert called black art. In other words, there are several issues that could be considered when using *Stacking*, such as the following:

- The algorithms that are used to create the base-level classifiers and their learning parameters,
- The number of base classifiers,
- The algorithm used to generate the meta-classifier and its learning parameters, and
- The type of attributes that should be used to create the meta-data.

One of the conclusions of this study is that there are many contradictory results and that there is no consensus on which Stacking configuration is optimal. This conclusion corroborates Wolpert's statement regarding the need for prior knowledge to configure these parameters.

Nevertheless, in recent years, there has been a trend in the literature toward Stacked Generalization, which is the use of meta-heuristics, such as genetic algorithms, ant colonies or artificial bee colonies, to automatically configure the *Stacking* system parameters. Thus, the *Stacking* system that is generated is domain dependent. However, this type of approach has a higher computational cost than other Stacking approaches because several generations of individuals must be evaluated to obtain the final system. Even if this task is not crucial for a large number of domains, given that most classification tasks do not require real-time operation, it could be a relevant issue in the era of big data. However, it would be interesting to explore adding incremental capabilities to *Stacking* in future research.

Although *Stacking* is applied to real-world problems less frequently than other ensemble methods, such as *Bagging* or *Boosting*, the exponential growth of data as well as the diversity of these data continues to make *Stacking* an interesting alternative for generating ensembles.

ACKNOWLEDGMENTS

This research was supported by the Spanish MICINN under projects TRA2010-20225-C03-01 and TRA 2011-29454-C03-03.

REFERENCES

1. Rumelhart DE, McClelland JL. *Parallel Distributed Processing*. The MIT Press; 1988.•
2. Quinlan J. Induction of decision trees. *Mach Learn* 1986, 1:81–106.
3. Michalski R. A theory and methodology of inductive learning. In: Michalski R, Carbonell J, Mitchell T, eds. *Machine Learning*. Berlin Heidelberg: Springer; 1983, 83–134.
4. Mitchell TM. *Machine Learning*. New York: McGraw Hill; 1997.
5. Ranawana R. Multi-classifier systems: review and a roadmap for developers. *Int J Hybr Intell Sys* 2006, 3:35–61.
6. Saitta L. Integrated architectures for machine learning. *Mach Learn Appl Lect Not Comput Sci* 2001, 2049:218–229.
7. Polikar R. Ensemble based systems in decision making. *IEEE Circuits Syst Mag* 2006, 6:21–45.
8. Dietterich TG. Ensemble methods in machine learning. *Multiple Classifier Sys Lect Notes Comput Sci* 1857, 2000:1–15.
9. Dietterich TG. Machine-learning research: four current directions. *AI Magazine* 1997, 18:97–137.
10. Kuncheva LI, Whitaker CJ. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Mach Learn* 2003, 51:181–207.
11. Breiman L. Bagging predictors. *Mach Learn* 1996, 24:123–140.
12. Schapire RE. The strength of weak learnability. *Mach Learn* 1990, 5:197–227.
13. Ho TK. The random subspace method for constructing decision forests. *IEEE Trans Pattern Anal Mach Intell* 1998, 20:832–844.
14. Optiz D. Feature selection for ensembles. In: *Proceedings of the 16th International Conference on Artificial Intelligence* 1999, pp. 379–384.•

15. Oliveira LS, Sabourin R, Bortolozzi F, Suen CY. A methodology for feature selection using multi-objective genetic algorithms for handwritten digit string recognition. *Int J Pattern Recogn Artif Intell* 2003, 17:903–929.

16. Tsymbal A, Pechenizkiy M, Cunningham P. Diversity in search strategies for ensemble feature selection. *Inform Fusion* 2005, 6:83–98.

17. Bryll R, Gutierrez-Osuna R, Quek F. Attribute bagging: improving accuracy of classifier ensembles by using random feature subsets. *Pattern Recogn* 2003, 36:1291–1302.

18. Anand R, Mehrotra KG, Mohan CK, Ranka S. An improved algorithm for neural network classification of imbalanced training sets. *IEEE Trans Neural Netw* 1993, 4:962–969.

19. Hastie T, Tibshirani R. Classification by pairwise coupling. *Ann Stat* 1998, 6:451–471.

20. Murphey YL, Wang H, Ou G. OAHO: an effective algorithm for multi-class learning from imbalanced data. In: *Proceedings of International Joint Conference on Neural Networks*; 2007.

21. Dietterich TG, Bakiri G. Solving multiclass learning problems via error-correcting output codes. *J Artif Intell Res* 1995, 2:263–286.

22. Sesmero MP, Alonso-Weber JM, Gutiérrez G, Ledezma A, Sanchis A. A new artificial neural network ensemble based on feature selection and class recoding. *Neural Comput Appl* 2012, 21:771–783.

23. Sesmero MP, Alonso-Weber JM, Gutierrez G, Ledezma A, Sanchis A. An ensemble approach of dual base learners for multi-class classification problems. *Inform Fusion* 2015, 24.•

24. Sesmero MP, Alonso-Weber JM, Gutiérrez G, Sanchis A. CCE: an approach to improve the accuracy in ensembles by using diverse base learners. In: Polycarpou M, de Carvalho APLF, Pan J-S, Woźniak M, Quintian H, Corchado E, eds. *Hybrid Artificial Intelligence Systems*, vol. 8480. Springer International Publishing; 2014, 630–641.

25. Kolen JF, Pollack JB. Backpropagation is sensitive to initial conditions. *Complex Syst* 1990, 4:269–280.

26. Dietterich TG. An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization. *Mach Learn* 2000, 40:139–157.

27. Wolpert D. Stacked generalization. *Neural Netw* 1992, 5:241–259.

28. Kuncheva LI. Switching between selection and fusion in combining classifiers: an experiment. *IEEE Trans Syst Man Cybern* 2002, 32:146–156.

29. Zhu X, Wu X. Dynamic classifier selection for effective mining from noisy data streams. In: *Proceedings of Fourht IEEE International Conference on Data Mining. ICDM'04*. 2004, 305–312.

30. Rokach L. Ensemble-based classifiers. *Artif Intell Rev* 2010, 33:1–39.

31. Chan P, Stolfo S. Toward parallel and distributed learning by meta-learning. *AAAI Workshop in Knowledge Discovery in Databases* 1993:227–240.•

32. Chan P, Stolfo S. On the accuracy of meta-learning for scalable data mining. *J Intell Inform Sys* 1997, 25:1–25.

33. Seewald A, Fürnkranz J. An evaluation of grading classifiers. *Adv Intell Data Anal Lect Notes Comput Sci* 2001, 2189:115–124.

34. Skalak DB. *Prototype selection for composite nearest neighbor classifiers*. 1997.•

35. Fan D, Stolfo S, Chan P. Using conflicts among multiple base classifiers to measure the performance of stacking. In: *Proceedings of the ICML-99 Workshop on Recent Advances in Meta-Learning and Future Work* 1999, pp. 10–17.

36. Merz C. Using correspondence analysis to combine classifiers. *Mach Learn* 1999, 36:33–58.

37. Ting K, Witten I. Issues in stacked generalization. *J Artif Intell Res* 1999, 10:271–289.

38. Breiman L. Stacked regressions. *Mach Learn* 1996, 24:49–64.

39. LeBlanc M, Tibshirani R. Combining estiamates in regression and classification. *J Am Stat Assoc* 1996, 91:1641–1650.

40. Seewald A. How to make stacking better and faster while also taking care of an unknown weakness. In: *Proceedings of the Nineteenth International Conference in Machine Learning* 2002.

41. Todorovski L, Džeroski S. Combining multiple models with meta decision trees. *Principles of Data Mining and Knowledge Discovery Lect Notes Comput Sci* 1910, 2000:54–64.

42. Džeroski S, Ženko B. Stacking with multi-response model trees. *Multiple Classifier Systems Lect Notes Comput Sci* 2002, 2364:201–211.

43. Menahem E, Rokach L, Elovici Y. Troika – an improved stacking schema for classification tasks. *Inform Sci* 2009, 179:4097–4122.

44. Ledezma A, Aler R, Borrajo D. Heuristic search-based stacking of classifiers. *Heuristics Optim Knowl Disc* 2002:54–67.•

45. Chen Y, Wong ML. An ant colony optimization approach for stacking ensemble. In: *Second World Congress on Nature and Biologically Inspired Computing* (*NaBIC*) 2010, pp. 146–151.

46. Chen Y, Wong ML. Optimizing stacking ensemble by an ant colony optimization approach. In: *13th Annual Genetic and Evolutionary Computation Conference, GECCO 2011*. Dublin, Ireland, 2011.

47. Shunmugapriya P, Kanmani S. Optimization of stacking ensemble configurations through artificial bee colony algorithm. *Swarm Evol Comput* 2013, 12:24–32.

48. Chan P, Stolfo S. A comparative evaluation of voting and meta-learning on partitioned data. In: *Proceedings of Twelfth International Conference on Machine Learning* 1995, pp 90–98.
49. Schaffer C. Cross-validation, stacking and bi-level stacking: methods for classification learning. In: Cheeseman P, Oldford W, eds. *Selecting Models from Data: Artificial Intelligence and Statistics IV*. Springer-Verlag; 1994, 51–59.
50. Ting KM. Decision combination based on the characterisation of predictive accuracy. *Intell Data Anal* 1997, 1:181–205.
51. Gama J, Brazdil P. Cascade generalization. *Mach Learn* 2000, 41.●
52. Torres-Sospedra J, Hernández-Espinosa C, Fernández-Redondo M. Combining MF networks: a comparison among statistical methods and stacked generalization. In: Schwenker F, Marinai S, eds. *Artificial Neural Networks in Pattern Recognition*, vol. 4087. Berlin Heidelberg: Springer; 2006, 210–220.
53. Cohen WW, Carvalho VR. Stacked sequential learning. In: *International Joint Conference on Artificial Intelligence*, 2005, 671–676.
54. Dietterich T. Machine learning for sequential data: a review. In: Caelli T, Amin A, Duin RW, de Ridder D, Kamel M, eds. *Structural, Syntactic, and Statistical Pattern Recognition*, vol. 2396. Berlin Heidelberg: Springer; 2002, 15–30.
55. Ledezma A, Aler R, Sanchis A, Borrajo D. GA-stacking: evolutionary stacked generalization. *Intell Data Anal* 2010, 14:89–119.
56. Ordoñez FJ, Ledezma A, Sanchis A. Genetic approach for optimizing ensemble of classifiers. In: *21th International Florida Artificial Intelligence Research Society Conference - FLAIRS 2008*. Florida, USA, AAAI Press, 2008.
57. Gatta C, Puertas E, Pujol O. Multi-scale stacked sequential learning. *Pattern Recogn* 2011, 44:2414–2426.
58. Lafferty J, McCallum A. *Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data*. MA, USA: ; 2001.●
59. Puertas E, Escalera S, Pujol O. Generalized multi-scale stacked sequential learning for multi-class classification. *Pattern Anal Appl* 2013.●
60. Trivedi B, Kapadia N. Modified stacked generalization with sequential learning. *Int J Comput Appl* 2012, 13:38–43.
61. Doumpos M, Zopounidis C. Model combination for credit risk assessment: a stacked generalization approach. *Ann Oper Res* 2007, 151:289–306.
62. Hu MY, Tsoukalas C. Explaining consumer choice through neural networks: the stacked generalization approach. *Eur J Oper Res* 2003, 146:650–660.
63. Qian B, Rasheed K. Stock market prediction with multiple classifiers. *Appl Intell* 2006, 26:25–33.
64. Qian B, Rasheed K. Foreign exchange market prediction with multiple classifiers. *J Forecasting* 2010, 284:271–284.
65. Töscher A, Jahrer M, Bell RM. The bigchaos solution to the netflix grand prize. *Netflix s* 2009:1–52.●
66. Sill J, Takács G, Mackey L, Lin D. Feature-weighted linear stacking. 2009:1–17, arXiv preprint arXiv:0911.0460.●
67. Netflix Prize homepage. Available at: http://www.netflixprize.com.
68. Escalera S, Puertas E, Radeva P, Pujol O. Multi-modal laughter recognition in video conversations. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2009. *CVPR Workshops 2009*, 2009.
69. Li R, Hapfelmeier A, Schmidt J, Perneczky R, Drzezga A, Kurz A, Kramer S. A case study of stacked multi-view learning in dementia research. *Artif Intell Med Lect Notes Comput Sci* 2011, 6747:60–69.
70. Tzanis G, Berberidis C, Vlahavas I. StackTIS: a stacked generalization approach for effective prediction of translation initiation sites. *Comput Biol Med* 2012, 42:61–69.
71. Wang Y, Witten I. *Induction of model trees for predicting continuous classes*. 1996.●
72. Todorovski L, Dzeroski S. Combining multiple models with meta decision trees. In: *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*, 2000, pp. 54–64.
73. Dzeroski S, Zenko B. Is combining classifiers better than selecting the best one? *Mach Learn* 2004, 54:255–273.

FURTHER READING

Kuncheva LI. *Combining Pattern Classifiers: Methods and Algorithms*. 2nd ed. John Wiley & Sons: 2014.●

Zhou Z. *Ensemble Methods: Foundations and Algorithms*. Taylor & Francis Group: 2012.

