# Combining Classifiers: from the creation of ensembles to the decision fusion

Moacir P. Ponti Jr.
Institute of Mathematical and Computer Sciences (ICMC)
University of São Paulo (USP) at São Carlos, SP, Brazil

Email: moacir@icmc.usp.br, Web: http://www.icmc.usp.br/~moacir

Abstract—Multiple classifier combination methods can be considered some of the most robust and accurate learning approaches. The fields of multiple classifier systems and ensemble learning developed various procedures to train a set of learning machines and combine their outputs. Such methods have been successfully applied to a wide range of real problems, and are often, but not exclusively, used to improve the performance of unstable or weak classifiers. In this tutorial are presented the basic terminology of the field, a discussion on the effectiveness of combination algorithms, the diversity concept, methods for the creation of an ensemble of classifiers, approaches to combine the decisions of each classifier, the recent studies and also possible future directions.

 ${\it Keywords} ext{-}{\it Classifier}$  Combination, Multiple Classifier Systems, Ensemble Learning.

#### I. Introduction

The general idea of combining pattern classifiers can be summarized by the use of a methodology to create an ensemble of learners and to produce a final decision given the outputs of those learners. This kind of approach is intuitive since it imitates our nature to seek several opinions before making a crucial decision [1].

The combination of expert opinions is a topic studied since the second half of the twentieth century. In the beginning the studies were devoted to applications such as democracy, economic and military decisions. Maybe the first learning model of a system with multiple experts is the Pandemonium architecture, described by Oliver Selfridge in 1959 [2]. Afterwards some important studies were published using different terms such as committee, classifier fusion, combination, aggregation, mixture of experts and others to indicate sets of learners that work in cooperation to solve a pattern recognition problem [3], [4], [5]. Nowadays the most common terms are "ensemble learning" and "multiple classifier systems" mostly used by the community of machine learning and pattern recognition, respectively.

The research field of "multiple classifier systems" (MCS) become very popular after the half of the 1990 decade, with many papers published on the creation of ensembles of classifiers that provided some theoretical insights of why combining classifiers could be interesting [6]. It was also disseminated by a review on statistical pattern recognition in 2000 [5]. By that time, a international workshop on MCS was created to assess the state of the art and the potential market of the field. Today

there are hundreds of papers about combination of classifiers and its applications and books dedicated to the subject.

Classifier ensemble is a set of learning machines whose decisions are combined to improve performance of the pattern recognition system. Much of the efforts in classifier combination research focus on improving the accuracy of difficult problems, managing weaknesses and strenghts of each model in order to give the best possible decision taking into account all the ensemble. The use of combination of multiple classifiers was demonstrated to be effective, under some conditions, for several pattern recognition applications.

Good and bad patterns and scenarios for combination were also experimentally and theoretically studied in the past years [7]. Many studies showed that classification problems are often more accurate when using combination of classifiers rather than an individual base learner. For instance "weak" classifiers are capable of outperform a highly specific classifier [8]. These methods were widely explored, for example, to stabilize results of random classifiers and to improve the performance of weak ones. Neural-network based classifiers that are unstable can be stabilized using MCS techniques [9]. Also, noisy data can be better handled since the diversity of classifiers included in the ensemble increases the robustness of the decisions [10]. Besides, there are many classifiers with potential to improve both accuracy and speed when used in ensembles [11]. All these advantages can be explored by researches on the field of pattern recognition and machine learning.

An introduction of the terminology, motivations and issues of ensemble methods are discussed in section II. Afterwards, we describe the methods of the two main steps of the design of a multiple classifier system: the creation of ensembles in section III, and the combination of the decisions in section IV. Final remarks and future directions on the field are given in section V.

#### II. WHY USE A MULTIPLE CLASSIFIER SYSTEM?

"No Free Lunch" theorems have shown that there is not a single classifier that can be considered optimal for all problems [12]. There is no clear guideline to choose a set of learning methods and it is rare when one has a complete knowledge about data distribution and also the about the details of how the classification algorithm behaves. Therefore, in practical pattern classification tasks it is difficult to find a good single classifier.

The choice of a single classifier trained with a limited (size or quality) dataset can make the design even more difficult. In this case, selecting the best current classifier can led to the choice of the worst classifier for future data. Specially when the data used to learn was not sufficiently representative in order to classify properly new objects, the test set provides just apparent errors  $\hat{E}$  that differ from true errors E, in a generalization error:  $\hat{E} = E \pm \Delta$ . This common situation, where small and not representative data is used as an input to a classifier, can led to difficulties when one must choose from a set of possible methods.

According to Dietterich [13], there are three main motivations to combine classifiers, the worst case, the best case and the computational motivation:

- Statistical (or worst case) motivation: it is possible to avoid the worst classifier by averaging several classifiers. It was confirmed theoretically by Fumera and Roli in 2005 [14]. This simple combination was demonstrated to be efficient in many applications. There is no guarantee, however, that the combination will perform better than the best classifier.
- Representational (or best case) motivation: under particular situations, fusion of multiple classifiers can improve the performance of the best individual classifier. It happens when the optimal classifier for a problem is outside the considered "classifier space". There are many experimental evidences that it is possible if the classifiers in an ensemble makes different errors. This assumption has a theoretical support in some cases when linear combination is performed.
- Computational motivation: some algorithms performs an optimization task in order to learn and suffer from local *minima*. Algorithms such as the backpropagation for neural networks are initialized randomly in order to avoid locally optimum solutions. In this case it is a difficult task to find the best classifier, and it is often used several (hundreds or even thousands) initializations in order to find a presumable optimal classifier. Combination of such classifiers showed to stabilize and improve the best single classifier result [9].

The bias-variance decomposition [15] is used on the studies that tries to understand better the theoretical side of the performance of ensembles. Tumer and Ghosh [16], based on manipulation used in bias-variance decomposition, provided a framework for analyzing the simple averaging combination rule, saying that the ensemble error will be just equal to the average error of the individuals if the classifiers are correlated, and will have an error M times smaller than the error of the individuals if the classifiers are statistically independent. The combination of low biased classifiers with high variance can reduce the variance, and the combination of low variance classifiers allow a bias reduction.

Other motivations are related to: i) applications that can naturally use a set of learners such as in sensor fusion, ii) the difficulty on the design of a pattern classifier by tuning

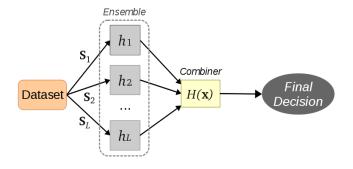


Fig. 1. Parallel architecture

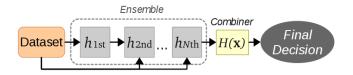


Fig. 2. Serial architecture

parameters, and iii) the availability of classifiers that exhibit different competences in different feature subspaces.

#### A. MCS Architectures and Taxonomies

Since MCS is an area that received contribution from several fields (mainly pattern recognition, machine learning and data fusion), various terms were used on the same notion. Although some are not absolutely identical, we will assume that: classifier = hypothesis = learner = expert = model, and that: example = instance = case = data point = object = pattern [6].

Let  $h_i$  be the  $i^{th}$  classifier on an ensemble of L classifiers,  $S = (x_1, y_1), (x_2, y_2), ..., (x_N, y_N)$  the training data set,  $S_i$  a version of the available dataset, and H(x) a function that combines/selects the decisions of various learners about a new input pattern x. It is possible to classify the systems into two main architectures:

- Parallel: with a representation showed in Figure 1, the set of classifiers are trained in parallel, and their output are combined afterwards to give the final decision. More general methodologies were developed using this architecture because it is simpler and easier to analyze.
- Serial: with a representation in Figure 2, a primary classifier is used, and when it is not able to classify some new pattern by rejecting it, we use a second classifier that is trained in order to be accurate on the errors of the previous classifier. A third and fourth classifier can be used and so on. Sequential methods are often application-specific, and are also useful in the context of on-line learning.

There are two main methods for the design of a MCS: the design with focus on creating a good classifier ensemble, and the design of the combination or selection method to obtain an optimal decision given some fixed classifiers [17]:

- Coverage optimization methods: create a good set of classifiers with characteristics that allow the use of a simple combinator. It is interesting for example when the creation of "strong" classifiers is difficult or time consuming. Methods to create such classifiers are discussed in section III.
- Decision optimization methods: the classifiers are given and are unchangeable, so the goal is to optimize the combination itself. In such cases a more complex combiner or selection method must be used. It is useful for methods that are specific to some problems or when the designer knowledge is sufficient to construct such classifiers.

# B. Diversity and Ensemble Performance

The intuition says that the classifiers of an ensemble should be at the same time as accurate as possible and as diverse as possible. It is known that the classifiers must be accurate, i.e, produce an error rate that is better than random guessing, on new input [18]. However "diversity" is an elusive concept [19]. Defining this concept and how it influences the performance is not trivial. It can, for example, depend both on the base classifiers and the complexity of the method used to obtain the final decision. The classifier diversity is under discussion. Good material on this topic can be found in the following references [19], [6], [7].

A point of consensus is that when the classifiers make statistically independent errors, the combination has the potential to increase the performance of the system. In order to understand better this idea, we can classify diversity in levels:

- 1) no more than one classifier is wrong for each pattern
- 2) the majority is always correct
- 3) at least one classifier is correct for each pattern
- 4) all classifiers are wrong for some patterns

If the designer knowledge about the diversity is good, it is possible to choose better the fusion method. For example, it is known that simple fusers can be used for classifiers with a simple complementary pattern, but more complex ones are necessary for classifiers with a complex dependency model.

How the diversity measures are related to the accuracy of the ensemble is a matter of ongoing research. The experimental studies observed the expected results. However, they also showed that the properties of an ensemble that are desired to obtain a successful combination are not common in practice.

A didactic example is shown in Figure 3, where 3 classifiers (assuming individual accuracies of 0.6) have to classify 10 unknown objects. The figure shows four cases: a statistically independent case, a case with identical classifiers and two cases of dependent classifiers. By this example it is clear that the independence contributed to a performance increase. When dependence is assumed, we can have incorrect or correct classification dependency, giving very different results.

Assessing the diversity of a classifier is also not a trivial task, since the concept itself is not clear, but there are pairwise measures that can be used. Consider for example the first two classifiers of the independent case of Figure 3. The number of samples both classified correctly was a=3/10=

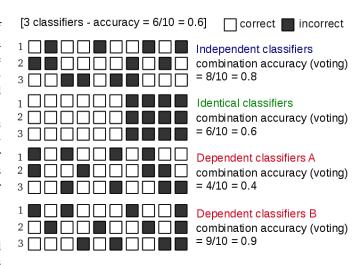


Fig. 3. Ensemble diversity example. Four cases indicates the possible performance, showing that independent classifiers potentially increases the individual performance, and two cases of dependent classifiers with opposite results.

0.3, the number of samples misclassified by the first one is b=4/10=0.4, by the second c=4/10=0.4, and, finally, the ones both misclassified was d=1/10=0.1. Note that a+b+c+d=1.

In the scenario described above, there are several pair-wise measures in the literature both inside and outside the context of classifier combination. The simplest one is the double fault measure that uses just d as reference. A commonly used method is the Q statistic [6]:

$$Q = \frac{ad - bc}{ad + bc},\tag{1}$$

that outputs a number in the [1, -1] interval, where lower numbers means higher diversity. There is also a similar measure called the inter-rated agreement [20]:

$$k = \frac{2(ad - bc)}{(a+c)(c+d) + (a+b)(b+d)}.$$
 (2)

In the example, after computing the three measures we have a double fault of d=0.1, a value of Q=-0.68 and a k=-0.37. Calculate the diversity of a dependent classifier using the same measures is left as an exercise.

Once the diversity is estimated, it is possible to design better the system. For instance, in general simple average or majority voting are optimal for classifiers for approximate accuracies and same pairwise correlation, whereas weighted average is preferred for ensemble of classifiers with different accuracy or different pair-wise correlations [14].

# III. CREATION OF ENSEMBLES

Many approaches are possible in order to create classifiers that are good to combine (diverse and accurate). Among the most popular methods are:

• using knowledge about the problem: heuristic method that can produce good results when the knowledge of the

designer is sufficient to build a tuned system regarding architecture and parameters to create diverse classifiers. It is used specially in natural multiple classifier systems such as for multiple sensors, and also when different representations of patterns are possible (statistical and structural).

- randomization: a combination of several classifiers trained using different random instances. It is useful for classifiers that depends on some random choices such as decision trees on the test of each internal node when random selection on n best tests are performed, or on neural network classifiers with random weight initialization. An example are the Random Trees [21].
- varying methods: in absence of a deeper knowledge, vary classifier methods, architectures and parameters until reach a desired performance.
- **training data manipulation**: training an ensemble of classifiers using different training sets by splitting [11], by using cross-validation ensembles (used with small subsets) [13], bagging [22] and boosting [23].
- input features manipulation: training an ensemble of classifiers using different subspaces of the feature space, such as in the random subspace method [24].
- output features manipulation: each component classifier is used to solve a subset of N classes. A simple example is to use two classifiers for a two-class problem in a one against all strategy. It is necessary to have a method to recover the original N class problem. The most famous technique in this approach is the Error-Correcting Output Coding (ECOC) [25].

Next section addresses the problem of creating ensembles with limited size or imbalanced data sets. Afterwards, some important approaches of training set, input and output features manipulation are described: Bagging (section III-B), Adaboost (section III-C), Random Subspace Method (section III-D) and ECOC (section III-E).

# A. Creating ensembles with limited or imbalanced datasets

For small datasets noise injection can be used to increase the number of objects. The first studies just inserted Gaussian noise into the data set in order generate artificial objects, but this approach is dangerous and can even insert outliers. One possible method is to create new patterns only along the direction of the k-nearest neighbors of each pattern [26].

Another approach is to produce a cross-validation ensemble. Each individual classifier is trained with a smaller version of the training set, leaving k objects out (leave-k-out cross validation). For each next member of the ensemble we include the k objects that were out and remove another k objects, and so on. In this approach, each learner is trained with a different version of the training set with a high overlapping.

In case of imbalanced datasets, in which the classification categories are not approximately equally represented, one can undersample the majority class, or also create new artificial samples for the minority class using an approach called SMOTE (synthetic minority ovesampling technique) [27], that

obtained better results with a combination of oversampling the minority class and undersampling the majority class.

# B. Bagging

The Bagging technique (bootstrap aggregating) [22] is based on the idea that bootstrap samples of the original training set will present a small change with respect to the original training set, but sufficient difference to produce diverse classifiers. Each member of the ensemble is trained using a different training set, and the predictions are combined by averaging or voting. The different datasets are generated by sampling from the original set, choosing N items uniformly at random with replacement. See the basic method in Algorithm 1.

## Algorithm 1 Bagging

10:

end if

11: end for

**Require:** Ensemble size L, training set S of size N.

```
    for i = 1 to L do
    S<sub>i</sub> ← N sampled items from S, with replacement.
    Train classifier h<sub>i</sub> using S<sub>i</sub>.
    end for
    for each new pattern do
    if outputs are continuous then
    Average the decisions of h<sub>i</sub>, i = 1, ..., L.
    else if outputs are are class labels then
    Compute the majority voting of h<sub>i</sub>, i = 1, ..., L.
```

The probability of any object not being selected is  $p=(1-1/N)^N$ . For a large N, a single bootstrap (version of the training set) is expected to contain around 63% of the original set, while 37% are not selected.

This method showed two interesting points:

- Instances of an unstable classifier trained using different bootstraps can show significant differences,
- Variance reduction.

As cited before, the bias-variance decomposition is often used to handle discussions on ensemble performances. In this context, when the classifiers has small bias (errors) but high variance, bagging helps to reduce the variance [28]. According to Breiman [22] it works better with unstable models such as decision trees and neural networks, and can possible not work well with simple or stable models such as k-Nearest Neighbors.

After creating the ensemble you can use any combination method. However it was showed that simple linear combination using averaging or majority voting (elementary combiners, see section IV-A) is optimal [28].

The number of bagged classifiers should be selected by experimenting for each application. In the literature it is common to use from 50 to 100 classifiers, but when an applications cannot handle this amount of processing time, this number must be better chosen. A recent study showed that with 10 classifiers the error reduction is around 90%, because bagging allow a fast variance reduction as the number

of bagged classifiers is increased. Curiously, despite the simplicity and many successful applications, there are situations where Bagging converges withouth affecting variance [19], so it seems that this method is still not fully understood.

#### C. Adaboost

Adaboost (adaptive boosting) [23], tries to combine weak base classifier in order to produce an accurate "strong" classifier. It is a Boosting method, with many variations and derived methods in the literature. It is similar to bagging since the classifiers are built over different training sets. However it combines using a serial method and it is also a ensemble lerning method, not a general methodology to construct an ensemble.

The origins of the Boosting methods comes from the study of Schapire [29] that related the strongly learnable and weakly learnable problems, by showing that they were equivalent. The author showed that a weak model, performing only slightly better than random guessing, could be boosted into an arbitrarily accurate strong model. The Adaboost algorithm of Freund and Schapire [23] was the first practical Boosting method.

The method is an iterative process that builds an ensemble of classifiers. The algorithm trains classifier sequentially, a new model per round. At the end of each round, the misclassified patterns are weighted in order to be considered more important in the next round, so that the subsequent models compensate error made by earlier classifiers. The learning algorithm of the classifier used in Adaboost must allow the use of a weight for each training pattern.

The idea is to give higher weights to the patterns that are misclassified and, in the next iteration, try to construct a classifier capable of classify correctly these kinds of patterns.

Algorithm 2 shows the procedure. Note that Bagging uses uniform distribution for  $S_i$ , while Adaboost adapts a nonuniform one  $W_i$  over the elements of S. It also checks if the current classifier has at least a performance better than random guessing, that is, the error  $\varepsilon_i < 0.5$ . When this condition is not true, another classifier is trained or the iteration is terminated.

After each iteration, the distribution  $W_i$  is updated so that half the distribution mass are over the samples misclassified by  $h_i$ . For example, if the error is 0.15, the next classifier will devote 50% of effort in order to classify correctly the examples from that 15% wrongly classified samples, while the other objects are less emphasized:

$$\sum_{h_i(x_n) \neq y_n} W_{i+1}(n) = 0.5.$$
(3)

The exponential bound on the error rate is sequentially minimized by the update on W. The upper bound on the training error is guaranteed to decrease monotonically with L. It obeys the following bound on a data sample S drawn from the true distribution W:

$$P_{x,y\sim S}(yH(x)<0) \le \prod_{i=1}^{L} 2\sqrt{\varepsilon_i(1-\varepsilon_i)}.$$
 (4)

## Algorithm 2 Adaboost

1: for i = 1 to L do

2:

**Require:** Ensemble size L; training set S of size N, where  $y_i \in \{+1, -1\}$ ; initialize uniform distribution  $W_i$  over S.

Train classifier  $h_i$  using distribution  $W_i$ .

```
3: Compute \varepsilon_i \leftarrow P_{W_i}(h_i(x) \neq y).

4: if \varepsilon_i \geq 0.5 then

5: break

6: end if

7: \alpha_i \leftarrow \frac{1}{2} \ln \left( \frac{1-\varepsilon_i}{\varepsilon_i} \right)

8: W_{i+1} \leftarrow \frac{W_{i+1} \exp(-\alpha_i y_i h_i(x_i))}{Z_i},

9: where Z_i is a normalization factor to ensure that W_{i+1} is a valid distribution

10: end for

11: for each new pattern x do

12: H(x) \leftarrow \operatorname{sign} \left( \sum_{i=1}^{L} \alpha_i h_i(x) \right)

13: end for
```

There are developments on the bounds of generalization error of voting systems, using margin-based theory, in Schapire's study. Linkings between Boosting and the statistical learning theory were found [30] and the bounds holds for the Boosting framework and also for any other voting system. A paper with 70 pages was recently published in order to discuss the why and how Adaboost succeeds [31]. Therefore, Boosting is another method that seems simple, has potential to many applications, but it is still under study.

## D. Random Subspace Method (RSM)

The Random Subspace Method (RSM) [24] randomly selects an arbitrary number of subspace from the original feature space, and build a classifier on each subspace. This randomization should create classifiers that are complementary. The combination can be carried out by simple fixed rules. Experimental evidences showed that RSM works well with feature spaces with large feature sets and redundant features. It avoids the curse of dimensionality. The concepts of RSM can be related to the theory of stochastic discrimination of Kleinberg [32].

The algorithm projects each feature vector into a less dimensional subspace, by selecting m random components. Algorithm 3 shows the procedure.

RSM is similar to Bagging but instead of sampling objects, it performs a kind of feature sampling withouth replacement since it would be useless to include one feature more than once. Note that the number of features that is used to train each classifier,  $d_i$  can be different from one classifier to another, although it is often set to be uniform.

This method seems to work well for large feature sets with redundant features. It was also observed that it avoids the "curse" of dimensionality [33]. The key issue is how to define the dimensionality of the subspaces (variable d). For this there is no clear guideline, for each application experiments have to be carried out to understand the effect of d on the performance.

# Algorithm 3 Random Subspace Method

**Require:** Ensemble size L; training set S of size N, where the number of features is D; choose  $d_i$  to be the number of features to train each individual classifier, where  $d_i < D$ , for i = 1, ..., L.

```
1: for i=1 to L do 
2: S_i \leftarrow d randomly choosen features out of D, without replacement.
```

3: Train classifier  $h_i$  using  $S_i$ .

4: end for

5: for each new pattern do

if outputs are continuous then

7: Average the decisions of  $h_i$ , i = 1, ..., L.

8: **else if** outputs are are class labels **then** 

9: Compute the majority voting of  $h_i$ , i = 1, ..., L.

10: end if

11: end for

	$h_{_1}$	$h_{_2}$	$h_{_{\mathcal{3}}}$	$h_{_4}$
$\omega_{_{I}}$	1	0	0	0
$\omega_{2}$	0	1	0	0
$\omega_{_{\mathcal{I}}}$	0	0	1	0
$\omega_{_{4}}$	0	0	0	1

Fig. 4. Example of an ECOC with one against all strategy with four classifiers in a four classes problem

## E. Error-Correcting Output Coding (ECOC)

This is a technique developed by Dietterich and Bakiri [25] that uses manipulation of output features, in which each component classifier is used to solve a subset of N classes. It creates code words by using several binary classifiers.

The ECOC uses the idea to transform multiclass problems into various 2-class problems. Take, for example, a multiclass problem with 4 classes (a polychotomy):  $\Omega = \{\omega_1, \omega_2, \omega_3, \omega_4\}$ . It could also be solved by a dichotomy with:  $\Omega = \{(\omega_1 \cup \omega_2), (\omega_3 \cup \omega_4)\}$ , using parallel dichotomizers.

The standard approach is to use one class against all, that is,  $\omega_k$  against  $\omega_1 \cup ... \cup \omega_{k-1} \cup \omega_{k+1} \cup ... \cup \omega_c$ , where c is the number of classes. The ECOC is a matrix where the number of rows is equal to the number of classes, i.e, there is one *code word* per class, and the number of columns (bits) is equal to the number of classifiers, where each "bit" is provided by a classifier. The Figure 4 shows an example of the one versus all strategy for a problem with four classes and four classifiers, where each classifier is going to provide a decision about a different class.

To construct a good ECOC, one must try to design a set of code words that provides:

1) good separation between rows, i.e., as large as possible

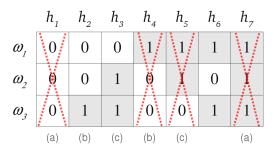


Fig. 5. Example of an ECOC with correlated columns (b), (c), and also useless bit settings on columns (a)

Hamming distance between any two code words, and

2) low correlation between columns, i.e., the classifiers should have different tasks to solve, so that diversity is created in the ensemble. See, for example.

Methods for constructing ECOC includes exhaustive codes, random assignment of bits, random assignment with even split, and based on a hierarchical structure of the classes [34].

This method is not well suited to problems with small number of classes. See an example of 3 classes on Figure 5, where some columns are considered correlated and can be removed from the ECOC, and also examples of useless bit settings.

When a new pattern arrives it is presented to all the classifiers in the ensemble. The decisions of the binary classifiers are going to produce a code word. The object is assigned to the class corresponding to the closest code word in the designed ECOC using some distance, often the Hamming distance.

## IV. COMBINATION OF DECISIONS

After creating an ensemble, there are numerous methods for combining classifiers. The final decision can be obtained by two principal approches:

- 1) **Integration (or fusion)**: all classifiers contribute to the final decision, assuming competitive classifiers.
- Selection: one classifier is used to give the final decision to each pattern. It assumes that classifiers are complementary.

Also, the type of information produced by the members of an ensemble [35] can be classified in three levels:

- **Abstract**: each classifier outputs the class label for each input pattern.
- Rank: each classifier outputs a ranking list of possible classes for each input pattern.
- Measurement: each classifier outputs a score, probability or confidence level for each input pattern.

The ensembles constructed with classifiers of the three levels can be combined using fixed or trained rules, using fusion or selection methods. However, there are levels that are restricted to some methods.

The following sections presents the combiners most used in practice taking into account the three possible levels.

	$\omega_{_{I}}$	$\omega_{2}$	$\omega_{\scriptscriptstyle \mathcal{J}}$	$\omega_{_{4}}$
$h_{_1}$	0.1	0.5	0.2	0.1
$h_2$	0.3	0.3	0.3	0.1
$h_3$	0.2	0.0	0.8	0.0

Rule				1	Result
Minimum	0.1	0.0	0.2	0.0	$\omega_{\scriptscriptstyle 3}$
Maximum	0.3	0.5	0.8	0.1	$\omega_{\scriptscriptstyle 3}$
Product	0.01	0.00	0.05	0.00	$\omega_{\scriptscriptstyle 3}$
Average	0.2	0.3	0.4	0.1	$\omega_{\scriptscriptstyle 3}$
Median	0.2	0.3	0.3	0.1	tie
Majority Vote	1	2	2	0	tie

Fig. 6. Fixed rules combiners in a problem with four classes and an ensemble of three classifiers with measurement outputs.

#### A. Elementary combiners

A theorectical framework for fixed rules was proposed by Kittler et. al [4]. It includes the sum, product, max, min, average and median rules. All these rules can be used to combine classifiers on measurement level:

- **minimum**: finds the *mininum* score of each class between the classifiers and assigns the input pattern to the class with the maximum score among the maximum scores.
- maximum: finds the maximum score of each class between the classifiers and assigns the input pattern to the class with the maximum score among the maximum scores.
- **product**: *multiplies* the score provided by each base classifiers and assigns the class label with the maximum score to given input pattern.
- **sum**: *adds* the score provided by each base classifiers and assigns the class label with the maximum score to given input pattern.
- average: finds the *mean* of the scores of each class between the classifiers and assigns the input pattern to the class with the maximum score among the means. It is equivalent to the sum rule.
- **median**: finds the *median* of the scores of each class between the classifiers and assigns the input pattern to the class with the maximum score among the medians.

Figure 6 shows an example of combination of classifiers on measurement level using fixed rules. The matrix where each cell represents the output measure of a classifier (row) about a class (column), is called *decision profile* (DP).

The **majority vote** is another elementary combiner that is discussed in separate since it works in the abstract level. Together with the average and the product rules, the majority vote is the most used combiner. As cited before, for several methods for creating ensembles the majority vote is the

#### TABLE I

Look-up table for BKS method. The column with bold values are the one that is consulted when the classifier ensemble outputs the labels (1,2,1)

Classifiers outputs $(h_1(x), h_2(x), h_3(x))$								
	1,1,1	1,1,2	1,2,1	1,2,2	2,1,1	2,1,2	2,2,1	2,2,2
$\omega_1$	100	50	76	89	54	78	87	5
$\omega_2$	8	88	17	95	20	90	95	100

optimal combiner.

To combine classifiers, the voting method finds what is the class output of each classifier and count it output as a vote for a class, and assigns the input pattern to the class with the majority vote. In order to avoid ties, the number of classifiers used for voting is usually odd.

A trainable variant is the weighted majority vote. The votes are multiplied by a weight that is often obtained by estimating the classifiers' accuracies on a validation set. One possible selection is:

$$w_i = \log\left(\frac{p_i}{1 - p_i}\right),\tag{5}$$

where  $p_i$  is the accuracy of the  $i^{th}$  classifier. This weight selection guarantees minimum error for the weighted majority vote when the outputs are independent. A similar approach can be used to produce a trainable the weighted average.

## B. Behaviour Knowledge Space (BKS)

In the Behaviour Knowledge Space method (BKS), every possible combination of abstract-level classifiers output is regarded as a cell in a look-up table. Each cell contains the number of samples of the validation set characterized by a particular value of class labels. It is a kind of brute force method to estimate the **posterior probabilities** using a multinomial model.

Table I shows an example of a look-up table. In this case, if a new pattern x is classified by the ensemble as  $(h_1(x),h_2(x),h_3(x))=(1,2,1)$ , we should look at the column corresponding to this pattern and choose the class with the maximum value (in this case class 1), since this value is related to the posterior probability of the classifier for each class,  $P(c=1|h_1(x)=1,h_2(x)=2,h_3(x)=1)=\frac{76}{76+17}=0.82$ .

A rejection option can be included in this method by defining a threshold on the computed posterior probability in order to limit error due to ambigous cells. In the above example 0.82 should be higher than a certain threshold to be accepted.

This method guarantees minimum error on the training set. It does not made assumptions regarding independence, but have some drawbacks since it does not works well with small data sets, and it is also severely affected by the curse of dimensionality.

#### C. Borda Count

The most common method to combine rank level classifiers is the borda count method. It **sums the rankings of each class** and assigns the new pattern to the class with the higher sum,

 ${\bf TABLE~II}\\ {\bf BORDA~COUNT~EXAMPLE~USING~THREE~CLASSIFIERS~AND~FOUR~CLASSES}$ 

Rank value	$h_1$	$h_2$	$h_3$
4	С	a	b
3	b	b	a
2	d	d	c
1	a	c	d

assuming that higher rank values means more confidence of the classifier.

Ranking methods such as Borda cound are suitable for multiclass problems where the correct class may appear in a high rank but sometimes not on the top. Rankings are also interesting to avoid problem of inconsistencies between continuous values provided by different classifiers, since normalization of classifier outputs (scores, confidences, posterior probabilities) is not trivial.

Let L=3, c=4 and  $\Omega=\{a,b,c,d\}$ , the Table II shows the rank value of the classes provided by each classifier for a given input pattern. The Borda Count method simply sums the rank for each class, resulting in:  $R_a=1+4+3=8$ ,  $R_b=3+3+4=10$ ,  $R_c=4+1+2=7$ ,  $R_d=2+2+1=5$ . In this example, the new pattern will be assigned to class b since it obtained the highest ranking sum, 10.

## D. Decision Templates

A decision profile (DP) is a matrix organization of the outputs of the L classifiers in an ensemble. Each cel i,j represents the support that classifier  $D_i$  gives to the hypotesys that the pattern comes from the class  $\omega_j$ , where i are rows and j are columns. This set of outputs can be considered an intermediate feature space, and can be represented as follows:

$$DP(x) = \begin{bmatrix} D_{1,1}(x) & \cdots & D_{1,c} \\ \cdots & \cdots & \cdots \\ D_{L,1}(x) & \cdots & D_{L,c} \end{bmatrix}, \tag{6}$$

where a column j can be seen as the support for the class  $\omega_j$ , and a row i is the output of a classifier  $D_i$  for each class.

Figure 6 shows a decision profile for an ensemble of three classifiers in a four class problem. The idea of the decision templates (DT) combiner [6] is to **capture the most typical decision profile for each class**, and classify new patterns by comparing their decision profiles with the DT of each class using some similarity measure. In order to train the method, for each class, calculate the mean of the decision profiles  $DP(s_k)$  of all members of  $\omega_j$  from the data set S. Call this mean a decision template  $DT_j$ :

$$DT_j = \frac{1}{N_j} \sum_{s_k \in \omega_j, s_k \in S} DP(s_k), \tag{7}$$

where  $N_j$  is the number of objects of S from the class  $\omega_j$ .

Given an input pattern x, construct a decision profile DP(x) using the outputs of the classifiers in the ensemble. Then, calculate the similarity S between DP(x) and each  $DT_j$ , obtaining a support  $\mu_j$  for each class:

$$\mu_j(x) = \mathcal{S}(DP(x), DT_j), j = 1, ..., c,$$
 (8)

where c is the number of classes. The final decision is made by selecting the class with the maximum support  $\mu_j$ . Using S as the squared Euclidian distance will be equivalent to applying the nearest mean classifier in the intermediate feature space. Another distance that comes from fuzzy set theory is the symmetric difference (DTS):

$$\mu_j(x) = 1 - \frac{1}{L \times c} \sum_{i=1}^{L} \sum_{k=1}^{c} \max\{a_{i,k}, b_{i,k}\}$$
 (9)

$$a_{i,k} = \min \left[ DT_i(i,k), (1 - d_{i,k}(x)) \right]$$
 (10)

$$b_{i,k} = \min\left[ (1 - DT_j(i,k)), d_{i,k}(x) \right]$$
(11)

The DTs are a simple, but more complex and interesting way, to combine outputs in measurement level. Another similar method is the Dempster-Shafer combination that also uses the decision templates, but a belief degree instead of a similarity measure in order to decide the class of a new input.

#### E. Stacked combination

The soft outputs of the L individual classifiers can be considered as features of a new classification problem (classifier output feature space). It is called stacked generalization approach by Wolpert [36].

Kuncheva [6] describes a simple example to illustrate the method. Let A, B, C and D be four disjoint subsets of the training data set. Suppose we have three classifiers  $h_1, h_2$  and  $h_3$  trained using a four-fold cross-validation process using A, B, C and D. At the end there will be four versions of each of our classifiers, trained respectively on (ABC), (BCD), (ACD) and (ABD). The combiner is trained on a data set of size N(points) obtained in the following way: for any data point in subset A we take the outputs for that point from the versions of  $h_1, h_2$  and  $h_3$  built on (BCD). In this way subset A has not been seen during the training of the individual classifiers. The three outputs togheter with the label of the new pattern form a data point in the training set for the combiner. It is repeated for the points from subset B, C and D. After the combiner has been trained, the classifiers are retrained with the whole training set. The new classifiers and the combiner are then ready for operation.

The disadvantages includes the fact that the dimensionality of the output space increases with the number of classes and classifiers. Also, the meta-classifier should be trained with a data set different from the one used for the individual classifiers. Besides, the characteristics of the resulting feature space are unknown. Finally, in small sample situations, "optimistically biased" experts' outputs can bluff the fusion rule (expert's boasting). Some discussions on experts' boasting in trainable fusion rules were addressed by Raudys [37].

#### F. Classifier selection

The objective of classifier selection methods is to select, for each pattern, a classifier that is able to correctly classify it. Selection methods are claimed to handle better correlated classifiers, among other advantages.

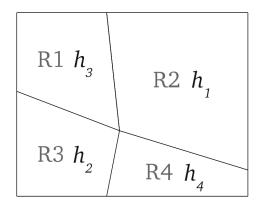


Fig. 7. Illustration of a partition of the feature space in four regions. Each classifier on the ensemble will be responsible for the decisions on each region.

One approach of classifier selection is called static selection and it works by dividing the feature space in regions. An illustrative example is shown in Figure 7. Patterns that belong to each region will be processed by a (single) different classifier. The regions can be found using a clustering method and, then, the accuracy of each classifier can be assessed inside each cluster so that each region (cluster) will be assigned to the "best" available classifier. Once a new input pattern arrives, the classifier assigned to the region of this new pattern is selected to produce the output [38].

Dynamic selection methods works by selecting a classifier for each new pattern. The Overall local accuracy (OLA) algorithm estimates the accuracies of classifiers in the local region of the new pattern, by assessing the percentage of the correctly labeled samples among the k nearest neighbors of the new pattern, and uses the decision of the more accurate classifier. Probabilistic methods such as the *a priori* or the *a posteriori* selection methods can also be used [39]. Other dynamic approaches are related to the studies on Adaptive Mixtures of Local Experts [3].

#### G. Fixed or trained combiners?

Some combination methods do not need training after the classifiers in the ensemble have been trained individually, for example the majority vote and the fixed rules. There are also methods that uses additional training, such as the weighted majority vote. A third class of methods develop the combiner during the training of the individual classifiers, for example the Adaboost.

According to Duin [40], several fixed rules depend only on the output values of the base classifiers, almost always suboptimal. The training set can be used to calibrate the base classifiers, and build a trained combined classifier. But how can one choose between such methods?

On the case of large data sets there are many possibilities, while for small data sets there are many challenges. When one must choose between fixed or trained combiners, it is important to observe the principal characteristics of each approach. A more detailed recommendation on this question is given by Kuncheva [6] and Duin [40].

Here, a summary of both approaches is described as a starting point. The use of fixed combiners:

- provides simplicity,
- · have low memory and time complexity,
- are suited for ensembles of classifiers with independent or low correlated error and similar performances.

On the other hand, trained combiners:

- provide potentially better performance than fixed rules,
- claimed to handle better correlated classifiers and classifier with different performances,
- have higher memory and time complexity,
- have heavy demands on the training set size and quality.

# V. FINAL REMARKS AND FUTURE DIRECTIONS

Classifier combination methods are a useful tool to consider when designing a solution to a pattern recognition problem. Theoretical and experimental studies showed the effectiveness of such methods and were able to provide some guidelines for the use of techniques to create ensembles and to combine the decisions. There are multiple approaches to randomize the ensemble parameters and to generate diverse models. As it is expected, there is not a best combiner that suits all problems, but one must choose the best combiner for a given dataset or problem. In this tutorial, some guidelines and the basics of the methods were presented in order to assist the reader on this choice.

Recent efforts and possible future directions of multiple classifier systems includes, among others: i) the development of more complex multi-sensor fusion applications such as multimodal biometrics, ii) the detection of anomalies, intrusions and frauds, with recent innovations in a field that is called *adversarial pattern recognition*, iii) the study of ensemble seletion, an extension of the classifier selection problem, and iv) methods that handle more complex problems and data, such as clustering ensembles. One important remark is the study of on-line learning, data streams, or *concept drift* in the context of multiple classifiers systems, is also receiving attention.

In order to implement the MCS techniques one can use the PRTools toolbox<sup>1</sup>, one of the most complete programming libraries including many ensemble methods.

To know more about classifier combination methods and discussion on how to measure diversity and performance of ensembles, the book of Kuncheva [6] is an excellent starting point. Another book was published by Rokach [1]. The most recent studies are usually presented on the series of the International Workshop on Multiple Classifier Systems, but can also be found in other major conferences in Machine Learning and Pattern Recognition areas

#### REFERENCES

- L. Rokach, Pattern Classification using Ensemble Methods. World Scientific, 2010.
- [2] O. Selfridge, "Pandemonium: A paradigm for learning," in *Proceedings of the Symposium on Mechanisation of Thought Processes*, D. V. Blake and A. M. Uttley, Eds., London, 1959, pp. 511–529.

<sup>1</sup>http://www.prtools.org

- [3] R. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Computation*, vol. 3, pp. 79–87, 1991.
- [4] J. Kittler, M. Hatef, R. Duin, and J. Matas, "On combining classifiers," *IEEE Trans Pattern Analysis and Machine Intelligence*, vol. 20, no. 3, pp. 226–239, 1998.
- [5] A. Jain, R. Duin, and J. Mao, "Statistical pattern recognition: a review," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 22, no. 1, pp. 4–37, 2000.
- [6] L. Kuncheva, Combining pattern classifiers: methods and algorithms. Wiley-Interscience, 2004.
- [7] G. Brown and L. I. Kuncheva, ""Good" and "bad" diversity in majority vote ensembles," in 9th Int. Work. on Multiple Classifier Systems (MCS 2010), LNCS 5997. Cairo, Egypt: Springer, 2010, pp. 124–133.
- [8] L. Kuncheva, J. Bezdek, and R. Duin, "Decision templates for multiple classifier fusion: an experimental comparison," *Pattern Recognition*, vol. 24, no. 2, pp. 299–314, 2001.
- [9] F. A. Breve, M. P. Ponti-Jr., and N. D. A. Mascarenhas, "Multilayer perceptron classifier combination for identification of materials on noisy soil science multispectral images," in *Proc. 20th Brazilian Symposium* on Computer Graphics and Image Processing (SIBGRAPI 2007). Belo Horizonte, MG: IEEE, Oct 2007, pp. 239–244.
- [10] M. P. Ponti-Jr. and N. D. A. Mascarenhas, "Material analysis on noisy multispectral images using classifier combination," in 6th IEEE Southwest Symposium on Image Analysis and Interpretation, 2004, pp. 1–5.
- [11] M. P. Ponti-Jr. and J. P. Papa, "Improving accuracy and speed of optimum-path forest classifier using combination of disjoint training subsets," in 10th Int. Work. on Multiple Classifier Systems (MCS 2011) LNCS 6713. Naples, Italy: Springer, 2011, pp. 237–248.
- [12] D. H. Wolpert, "The lack of a priori distinctions between learning algorithms," *Neural Computation*, vol. 8, no. 7, pp. 1341–1390, 1996.
- [13] T. G. Dietterich, "Ensemble methods in machine learning," in 1st Int. Work. on Multiple Classifier Systems (MCS 2000), LNCS 1857. Cagliary, Italy: Springer, 2000, pp. 1–15.
- [14] G. Fumera and F. Roli, "A theoretical and experimental analysis of linear combiners for multiple classifier systems," *IEEE Trans. Pattern Analysis* and Machine Intelligence, vol. 27, pp. 924–956, 2005.
- [15] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural Computation*, vol. 4, no. 1, pp. 1–58, 1992
- [16] K. Tumer and J. Ghosh, "Error correlation and error reduction in ensemble classifiers," *Connection Science*, vol. 8, no. 3,4, pp. 385–204, 1996
- [17] T. K. Ho, "Data complexity analysis for classifier combination," in 2nd Int. Work. on Multiple Classifier Systems (MCS 2001), LNCS 2096. Cambridge, UK: Springer, 2001, pp. 53–67.
- [18] L. Hansen and P. Salamon, "Neural network ensembles," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 12, no. 10, pp. 993–1001, 1990.
- [19] G. Brown, J. Wyatt, R. Harris, and X. Yao, "Diversity creation methods: a survey and categorisation," *Journal of Information Fusion*, vol. 6, no. 1, pp. 5–20, 2005.
- [20] D. Margineantu and T. G. Dietterich, "Pruning adaptive boosting," in Proc. 14th International Conference on Machine Learning (ICML-97). San Francisco, CA, USA: Morgan-Kaufmann, 1997, pp. 211–218.
- [21] L. Breiman, "Random forests," Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.
- [22] —, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

- [23] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proc. 13th International Conference on Machine Learning* (ICML-96), 1996, pp. 148–156.
- [24] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 832–844, 1998.
- [25] T. G. Dietterich and G. Bakiri, "Solving multiclass learning problems via error-correcting output codes," *Journal of Artificial Intelligence Research*, vol. 2, pp. 263–286, 1995.
- [26] M. Skurichina and R. P. W. Duin, "Bagging, boosting and the random subspace method for linear classifiers," *Pattern Analysis and Applica*tions, vol. 5, pp. 121–135, 2002.
- [27] N. Chawla, K. Bowyer, L. Hall, and W. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–257, 2002.
- [28] G. Fumera, F. Roli, and A. Serrau, "A theoretical analysis of bagging as a linear combination of classifiers," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 30, no. 7, pp. 1293–1299, 2008.
- [29] R. E. Schapire, "The strength of weak learnability," *Machine Learning*, vol. 5, pp. 197–227, 1990.
- [30] G. Rätsch, S. Mika, B. Schölkopf, and K. R. Müller, "Constructing Boosting algorithms from SVMs: an application to one-class classification," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 9, pp. 1184–1199, 2002.
- [31] D. Mease and A. Wyner, "Evidence contraty to the statistical view of boosting," *Journal of Machine Learning Research*, vol. 9, pp. 131–156, 2008.
- [32] E. M. Kleinberg, "On the algorithmic implementation of stochastic discrimination," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 5, pp. 473–490, 2000.
- [33] L. I. Kuncheva, F. Roli, G. L. Marcialis, and C. A. Shipp, "Complexity of data subsets generated by the random subspace method: an experimental investigation," in 3rd Int. Work. on Multiple Classifier Systems (MCS 2002), LNCS 2096. Cagliary, Italy: Springer, 2001, pp. 349–358.
- [34] O. Pujol, P. Radeva, and J. Vitri, "Discriminant ECOC: A heuristic method for application dependent design of Error Correcting Output Codes," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 28, no. 6, pp. 1007–1012, 2006.
- [35] L. Xu, A. Kryzak, and C. Y. Suen, "Methods of combining multiple classifiers and their applications to handwriting recognition," *IEEE Trans. Systems, Man and Cybernetics*, vol. 22, no. 3, pp. 418–435, 1992.
- [36] D. H. Wolpert, "Stacked generalization," Neural Networks, vol. 5, pp. 241–259, 1992.
- [37] S. Raudys, "Experts' boasting in trainable fusion rules," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 25, no. 9, pp. 1178–1182, 2003.
- [38] L. I. Kuncheva, "Clustering-and-selection model for classifier combination," in Proc. 4th Int. Conf. Knowledge-based Intelligent Engineering Systems and Allied Technologies. Bringhton, UK: IEEE, 2000, pp. 185–188.
- [39] L. Didaci, G. Giacinto, F. Roli, and G. L. Marcialis, "A study on the performances of dynamic classifier selection based on local accuracy estimation," *Pattern Recognition*, vol. 38, no. 11, pp. 2188–2191, 2005.
- [40] R. P. W. Duin, "The combining classifier: To train or not to train?" in *Proc. 16th Int. Conference on Pattern Recognition (ICPR 2002)*. Quebec, Canada: IEEE, 2002, pp. 765–770.