Information Sciences 179 (2009) 1298–1318

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

# Incremental construction of classifier and discriminant ensembles

Aydın Ulaş [a,*], Murat Semerci [b,1], Olcay Taner Yıldız [c], Ethem Alpaydın [a]

[a] Department of Computer Engineering, Boğaziçi University, TR-34342 Bebek, Istanbul, Turkey
[b] Department of Computer Science, Rensselaer Polytechnic Institute, 110 8th Street, Lally Hall, Troy, NY 12180-3590, USA
[c] Department of Computer Engineering, Işık University, TR-34980 Şile, Istanbul, Turkey

ARTICLE INFO

*Article history:*
Received 29 March 2008
Received in revised form 15 December 2008
Accepted 27 December 2008

*Keywords:*
Classification
Classifier fusion
Classifier ensembles
Stacking
Machine learning
Voting
Discriminant ensembles
Diversity

ABSTRACT

We discuss approaches to incrementally construct an ensemble. The first constructs an ensemble of classifiers choosing a subset from a larger set, and the second constructs an ensemble of discriminants, where a classifier is used for some classes only. We investigate criteria including accuracy, significant improvement, diversity, correlation, and the role of search direction. For discriminant ensembles, we test subset selection and trees. Fusion is by voting or by a linear model. Using 14 classifiers on 38 data sets, incremental search finds small, accurate ensembles in polynomial time. The discriminant ensemble uses a subset of discriminants and is simpler, interpretable, and accurate. We see that an incremental ensemble has higher accuracy than bagging and random subspace method; and it has a comparable accuracy to AdaBoost, but fewer classifiers.



## 1. Introduction

### 1.1. Current trends in classifier combination

It is well known that there is no single classification algorithm that is always most accurate, and methods have been proposed to combine classifiers based on different learning algorithms [20,31]. Each algorithm has a different inductive bias, that is, makes a different assumption about the data and makes errors on different instances and by suitable combination, the overall error can be decreased.

There are several methods for model combination: the simple method is to use voting which corresponds to the sum rule, or another fixed rule, i.e., median, product, minimum, or maximum [27]. Methods based on resampling from a single data set, such as bagging [8] and AdaBoost [16], are not used to combine different learning algorithms. In stacking, fusion is done using a trained, second layer classifier which estimates the real output from the outputs of base classifiers [58]. Ting and Witten [55] propose the Multiresponse Linear Regression (MLR) algorithm, which combines the outputs of base learners linearly. In a mixture of experts (MoEs) architecture, models are local and a separate gating network selects one of the local experts based on the input [24].

Model combination, however, is no panacea, and models in the ensemble should be carefully chosen for error to decrease. In particular, model combination through averaging reduces variance [8], and hence error, but only if bias does not increase

* Corresponding author. Tel.: +90 212 359 4523/4524; fax: +90 212 287 2461.
*E-mail address:* ulasmehm@boun.edu.tr (A. Ulaş).
[1] Tel.: +1 518 276 8326; fax: +1 518 276 4033.

0020-0255/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2008.12.024

in the process, or if the concomitant increase in the bias is small with respect to the decrease in the variance. It is therefore essential that only those models that contribute to accuracy are added and the poorly performing ones are weeded out.

### 1.2. Subset selection in classifier combination

Additional to its effect on statistical accuracy, each additional model increases space and computational complexity. A new model may also be sensing/extracting a costly representation which can be saved if the model is considered redundant. Methods therefore have been proposed to choose a small subset from a large set of candidate models. Since there are $2^L$ possible subsets of $L$ models one cannot try for all possible subsets unless $L$ is small, and various methods have been proposed to get a reasonable subset of size $m < L$ in reasonable time.

Ensemble construction methods also differ in the criterion they optimize. Additional to the methods that directly optimize ensemble accuracy, heuristics have also been proposed as measures of "diversity" in pinpointing models that best complement each other, to allow diverse ones to be added and similar ones to be deemed redundant and pruned.

Ensemble construction can be viewed as an optimization problem [64,50] and methods proposed in the literature correspond to different search strategies in optimization: there are greedy "forward" algorithms that are incremental, and add one model at a time if the addition improves the criterion that is to be optimized. There are "backward" search methods that prune from a large set if the removal is not harmful. There are also "floating" methods that do both, as well as ones that use genetic algorithms whose operators allow both addition and deletion. A chronological review of major ensemble construction methods in more detail is deferred to Section 6.

### 1.3. Proposed methods for ensemble construction

In this paper, we discuss and evaluate two ensemble construction approaches:

(1) We incrementally construct an *ensemble of classifiers* as in the methods discussed above. On 38 data sets using 14 different base classifiers, we test the effect of: (i) the criterion to be optimized (accuracy, statistically significant improvement and two diversity measures, correlation and $Q$ statistics), (ii) the search direction (forward, backward, floating), and (iii) the combiner (fixed voting, trained linear combiner).
(2) We incrementally construct an *ensemble of discriminants* where a classifier may be used for some of the classes but not for others. For example, in a three-class problem $(C_1, C_2, C_3)$, $C_1$ may be linearly separable from $C_2$ and $C_3$ and therefore the first discriminant of the linear classifier is chosen. If $C_2$ is not linearly separable from $C_1$ and $C_3$, the second discriminant of the linear classifier cannot be used and a more complicated classifier (e.g., $k$-nearest neighbor) needs to be incorporated in the ensemble.

This work has two goals:

(1) First, we investigate the effect of various factors in ensemble construction using a wide variety of learning algorithms, data sets and evaluation criteria. We compare this algorithm with bagging, boosting and random subspace method.
(2) Second, we generalize the idea of subset selection to the level of discriminants, to check if it is applicable not only at the level of classifiers but also at the level of discriminants that make up a classifier.

The paper is organized as follows: in Sections 2 and 3, we introduce our algorithms. We give the details of our experiments in Section 4 and results in Section 5. We discuss related work in Section 6 and conclude in Section 7.

## 2. Constructing an ensemble of classifiers

### 2.1. The Icon algorithm

Our proposed algorithm Icon to choose $m$ out of $L$ base classifiers is greedy in that it starts with the empty set and Incrementally CONstructs an ensemble where at each iteration, it chooses among all possible classifiers the one that best improves the performance when added to the current ensemble. The performance is measured using an ensemble evaluation criterion, as we will discuss next. The algorithm stops when there is no further improvement. Of course, this does not guarantee the finding of the best subset, but this algorithm has polynomial complexity, $O(L^2)$, whereas exhaustive searching of all possible subsets, $O(2^L)$, is of exponential complexity. As we see shortly, despite its simplicity, this algorithm works very well in practice.

The pseudocode of the algorithm is given in Fig. 1. We start with $E^{(0)} = \emptyset$. At iteration $t$ of the algorithm, we have ensemble $E^{(t)}$ containing $t$ models. Given the set of remaining $L - t$ candidate models, $M_k \notin E^{(t)}$, we have new candidate ensembles for iteration $t + 1$ as $S_k^{(t+1)} \equiv E^{(t)} \cup M_k,\ k = 1, \ldots, L - t$. Among these, we choose the one that is preferred to all other candidates and is also preferred to the current ensemble (Line 5 of Fig. 1).

```
1  function icon(P)
2  E^0 ← ∅
3  for t = 0 to L − 1
4      S_k^(t+1) ← E^(t) ∪ M_k, ∀M_k ∈ P where M_k ∉ E^(t)
5      if ∃S_j^(t+1) such that S_j^(t+1) ≺ S_k^(t+1), ∀k ≠ j
           and S_j^(t+1) ≺ E^(t)
6          then E^(t+1) ← S_j^(t+1), t ← t + 1
7          else break
8  end for
9  return E^(t)
```

**Fig. 1.** Pseudocode of the forward searching ICON algorithm.

$E_i \prec E_j$ denotes the binary relation of "preference" comparing two ensembles and holds if $E_i$ is preferred to $E_j$ according to the criterion used (which we will discuss next). If none of the candidates is preferred to $E^{(t)}$, the algorithm stops and $E^{(t)}$ is taken as the final ensemble.

The algorithm discussed above implements a forward, incremental, growing search. It is possible to implement a backward, decremental, pruning version, where one starts with the whole set and at each step removes one, again using the same preference relation and optimizing the same criterion. Similarly, a floating algorithm which tries to prune a previously added base classifier may also be envisaged.

### 2.2. Ensemble evaluation

Given two ensembles $E_i$ and $E_j$, we need an evaluation method to prefer one to the other. Note that this evaluation is done on a validation data set different from the data set, on which the base classifiers are trained. To average over randomness in splits, we also use cross-validation to generate multiple training/validation set pairs, train and validate multiple times and look at the paired validation values. We use the following ensemble evaluation criteria in our experiments:

(1) *Accuracy* (ACC): We compare the average accuracies of the two ensembles on the validation sets.
(2) *Cross-validation* (CV): We use a statistical test to check if the difference in accuracy is statistically significant, and allow only those additions that cause a statistically significant increase in accuracy. We use the one-sided $5 \times 2$ cv paired $t$-test [14] and check whether the more costly ensemble is significantly more accurate than the simpler one.
Note that cross-validation is only one method that can be used for model selection. One can also use a regularization approach to check whether a small increase in accuracy is worth an increase in complexity and use an augmented goodness measure:

$$\text{goodness}(E_i) = \text{accuracy}(E_i) - \lambda |E_i| \tag{1}$$

where $|E_i|$ measures the complexity of the ensemble which depends on the size of the ensemble as well as the time/space complexities of the base classifiers or the cost of sensing/preprocessing the inputs they use [12]. $\lambda$ gives the accuracy-complexity trade-off. Examples are AIC, BIC, and MML [21]. Equally, one can use a Bayesian approach where simpler models are assigned higher prior probabilities.
(3) *Q Statistic* (QSTAT): As a diversity measure, we use the $Q$ statistic [35]. First, we select the two classifiers which form the most diverse ensemble, and at each iteration, we add another classifier if diversity increases; else, we stop. The $Q$ statistic for two classifiers is calculated as

$$Q_{i,j} = \frac{N_{11}N_{00} - N_{10}N_{01}}{N_{11}N_{00} + N_{10}N_{01}} \tag{2}$$

where $N_{00}$, $N_{01}$, $N_{10}$, and $N_{11}$ are defined as in Table 1. The average diversity for an ensemble is calculated as

$$Q_{av} = \frac{2}{m(m-1)} \sum_{i=1}^{m-1} \sum_{k=i+1}^{m} Q_{i,k} \tag{3}$$

**Table 1**
Contingency table used by the measures of diversity.

| | $E_j$ correct | $E_j$ wrong |
|---|---|---|
| $E_i$ correct | $N_{11}$ | $N_{10}$ |
| $E_i$ wrong | $N_{01}$ | $N_{00}$ |

(4) *Correlation coefficient* (CORR): As another frequently used diversity measure, we use the average correlation coefficient [35]:

$$\rho_{i,j} = \frac{N_{11}N_{00} - N_{10}N_{01}}{\sqrt{(N_{11}+N_{10})(N_{01}+N_{00})(N_{11}+N_{01})(N_{10}+N_{00})}} \tag{4}$$

The average correlation for the ensemble is calculated by averaging over all pairs, as done in QSTAT.

### 2.3. Model combination

Given an ensemble of base classifiers, the easiest way to calculate the overall output is by taking a sum, which corresponds to taking a vote. Our base classifiers generate posterior probabilities so there is no need for scaling, normalization or transformation [36,25].

There are other fixed rules, i.e., median, product, minimum, or maximum [27], but the sum rule we use is known to work best in practice. Alkoot and Kittler [2] investigate fixed rules and see that the sum and the median rules are more robust to noise. Kuncheva [30] discusses six fixed rules for classifier combination on two-class problems with Gaussian and uniform error. She concludes that the min/max rule find the best ensemble when uniform error is the case. For Gaussian distributed errors, the combination rules behave similarly. Cabrera [9] analyzes average, median and maximum rules when the number of classifiers becomes large. He finds that average is the best for normal error, maximum is the best for uniform error and median is the best for Cauchy error. These studies analyze under the assumption that classifiers are independent, though in practice they are correlated [57].

Tax et al. [54] show that in multi-class problems, the product rule may be superior to the sum rule when the independent data representation assumption is met. On the other hand, the sum rule is more robust to noise. Especially in cases when one of the classifiers is an outlier, the veto power of the product rule decreases the combination performance. In two-class problems though, there is no difference.

The sum rule gives equal weight to all base classifiers, we also try a trained linear combiner; this is called stacking [58,15,45,46]. We do not constrain weights to be nonnegative or sum to 1, and there is also a constant intercept. A data set separate from the one used to train the base classifiers is used to train this linear combiner.

A trained rule may have lower bias, but fixed rules are generally favored for a number of reasons: (i) there is no extra cost of storing/processing the combiner model (called $L_1$ model in stacking), (ii) we save from the time needed to train the combiner model, and (iii) there is no need to leave out a part of the training set to train the combiner, and all data can be used to train the base classifiers.

## 3. Ensemble of discriminants

### 3.1. Rationale

The ICON algorithm above constructs an *ensemble of classifiers*, that is, it aims in finding a subset of base classifiers that best complement each other to maximize the overall accuracy. Now in this section, we consider each classifier as outputting a set of discriminants, and when we have a set of classifiers, we think of this as a larger set of discriminants, from which we choose a subset as before, constructing an *ensemble of discriminants*.

In the ensemble of classifiers, we choose $m$ classifiers from $L$ candidate classifiers. In the ensemble of discriminants where $k$ is the number of classes, we choose $p \geqslant k$ discriminants from $L \cdot k$ discriminants. We consider a new intermediate $L \cdot k$ dimensional space in which we do feature selection and then classification. This is a continuation of the idea of subset selection where the sole difference is that items that are selected are not whole classifiers but separate discriminants which make up the classifiers.

### 3.2. Discriminant selection

We use three algorithms for constructing a discriminant ensemble:

(1) *Forward subset selection* (FSS): This is a greedy, incremental algorithm that adds one discriminant at a time until there is no further improvement in accuracy; it is basically ICON adapted to discriminant selection. There is a linear combiner to calculate the overall output from the discriminants.
(2) *Decision tree* (DT): A decision tree is trained to learn the final output from the $L \cdot k$ dimensional discriminant values. Note that the decision tree acts both as a feature selector and as a classifier (i.e., combiner).
(3) DT *with linear output* (DT.LIN): A decision tree is trained as above but instead of using it also for decision, it is used for feature selection only. That is, we first train the tree, take the features it uses, and give them as input to a linear combiner.

### 3.3. Ensembles of discriminants vs. classifiers

The ensemble of discriminants is interesting for a number of reasons:

- A classifier may be accurate for some classes but not all, and it is best to use it only for the classes it is good at. Each classifier has a certain inductive bias, which may hold for some classes but not all. Take the case of a linear discriminant: In a multi-class case, one of the classes may be linearly separable from the others but another class may not be. The output of the linear classifier may therefore be included in the ensemble of discriminants for that class, but for some other class, a more complex discriminant should be included in the ensemble. Certain classes may be handled by a single discriminant, i.e., the output of one classifier suffices, but more discriminants are combined for more complex classes.
- When we know that a classifier will not be used for some classes, the overall complexity can be decreased, in terms of both memory and time. The parameters for the unused discriminants need not be stored and the unused discriminants need not be calculated. For example, when we are using support vector machines, we do not need to store the support vectors of a discriminant nor do the costly kernel calculations if it is not to be used.
- The base discriminants do not need to be posteriors or normalized in any manner. Both the linear model and the decision tree can use inputs in any scale, and no transformation or scaling is necessary thereby avoiding the risk of distorting values [36,25]. (Though in practice, normalizing the inputs may help convergence with linear models in certain situations.)
- The ensemble of discriminants by doing feature selection allows knowledge extraction. We know which base classifiers can be trusted for which classes. There may be certain base classifiers whose discriminants are not chosen for any class, there may be ones that are chosen for one or few, and there may be ones that are chosen for many. Certain classes may be

**Table 2**
Properties of data sets.

| Name | Source | # of instances | $k$ | $d$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Numerical | Categorical | | | | | | | | | | | | | | Total |
| | | | | | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 12 | 14 | 25 | 26 | 63 | |
| Australian | uci | 690 | 2 | 6 | 4 | 2 | | | | | 1 | | | | 1 | | | | 14 |
| Balance | uci | 625 | 3 | 0 | | | | 4 | | | | | | | | | | | 4 |
| Breast | uci | 699 | 2 | 9 | | | | | | | | | | | | | | | 9 |
| Bupa | uci | 345 | 2 | 6 | | | | | | | | | | | | | | | 6 |
| Car | uci | 1728 | 4 | 0 | | 3 | 2 | 1 | | | | | | | | | | | 6 |
| Cmc | uci | 1473 | 3 | 3 | | 2 | 4 | | | | | | | | | | | | 9 |
| Credit | uci | 690 | 2 | 6 | 4 | 3 | | | | | | 1 | | | 1 | | | | 15 |
| Cylinder | uci | 540 | 2 | 29 | 8 | 4 | | 2 | | 1 | 1 | | | | | | | | 45 |
| Dermatology | uci | 366 | 6 | 34 | | | | | | | | | | | | | | | 34 |
| *E. coli* | uci | 336 | 8 | 7 | | | | | | | | | | | | | | | 7 |
| Flags | uci | 194 | 8 | 26 | 13 | 2 | 2 | 1 | 1 | 1 | 3 | | 1 | 1 | 1 | | | | 26 |
| Flare | uci | 323 | 3 | 7 | | | 1 | | 2 | | | | | | | | | | 10 |
| Glass | uci | 214 | 6 | 9 | | | | | | | | | | | | | | | 9 |
| Haberman | uci | 306 | 2 | 3 | | | | | | | | | | | | | | | 3 |
| Heart | uci | 270 | 2 | 13 | | | | | | | | | | | | | | | 13 |
| Hepatitis | uci | 155 | 2 | 19 | | | | | | | | | | | | | | | 19 |
| Horse | uci | 368 | 2 | 23 | | 1 | | | | | 1 | | | | | | | 1 | 26 |
| Ionosphere | uci | 351 | 2 | 34 | | | | | | | | | | | | | | | 34 |
| Iris | uci | 150 | 3 | 4 | | | | | | | | | | | | | | | 4 |
| Monks | uci | 432 | 2 | 6 | | | | | | | | | | | | | | | 6 |
| Mushroom | uci | 8124 | 2 | 0 | 6 | 1 | 5 | 1 | 2 | 1 | | 4 | 1 | 1 | | | | | 22 |
| Nursery | uci | 12,958 | 4 | 0 | 1 | 3 | 3 | 1 | | | | | | | | | | | 8 |
| Optdigits | uci | 3823 | 10 | 64 | | | | | | | | | | | | | | | 64 |
| Pageblock | uci | 5473 | 5 | 10 | | | | | | | | | | | | | | | 10 |
| Pendigits | uci | 7494 | 10 | 16 | | | | | | | | | | | | | | | 16 |
| Pima | uci | 768 | 2 | 8 | | | | | | | | | | | | | | | 8 |
| Ringnorm | delve | 7400 | 2 | 20 | | | | | | | | | | | | | | | 20 |
| Segment | uci | 2310 | 7 | 19 | | | | | | | | | | | | | | | 19 |
| Spambase | uci | 4601 | 2 | 57 | | | | | | | | | | | | | | | 57 |
| Tae | uci | 151 | 3 | 1 | 2 | | | | | | | | | | | 1 | 1 | | 5 |
| Thyroid | uci | 2800 | 4 | 6 | 21 | | | | | | | | | | | | | | 27 |
| Tictactoe | uci | 958 | 2 | 0 | | 9 | | | | | | | | | | | | | 9 |
| Titanic | delve | 2201 | 2 | 0 | 2 | | 1 | | | | | | | | | | | | 3 |
| Twonorm | delve | 7400 | 2 | 20 | | | | | | | | | | | | | | | 20 |
| Vote | uci | 435 | 2 | 0 | 16 | | | | | | | | | | | | | | 16 |
| Wine | uci | 178 | 3 | 13 | | | | | | | | | | | | | | | 13 |
| Yeast | uci | 1484 | 10 | 8 | | | | | | | | | | | | | | | 8 |
| Zoo | uci | 101 | 7 | 16 | | | | | | | | | | | | | | | 16 |

easy in that one discriminant is enough; a number of discriminants need to be combined to learn more difficult classes accurately.

## 4. Experimental details

### 4.1. Data sets

We use a total of 38 data sets, where 35 of them (*zoo*, *iris*, *tae*, *hepatitis*, *wine*, *flags*, *glass*, *heart*, *haberman*, *flare*, *ecoli*, *bupa*, *ionosphere*, *dermatology*, *horse*, *monks*, *vote*, *cylinder*, *balance*, *australian*, *credit*, *breast*, *pima*, *tictactoe*, *cmc*, *yeast*, *car*, *segment*, *thyroid*, *optdigits*, *spambase*, *pageblock*, *pendigits*, *mushroom*, *and nursery*) are from UCI [4] and 3 (*titanic*, *ringnorm*, *and twonorm*) are from Delve [44] repositories. The properties of these data sets can be seen in Table 2.

### 4.2. Base classifiers

We use 14 base classifiers which we have chosen to span the wide spectrum of possible machine learning algorithms as much as possible:

(1–3) *knn*: $k$-nearest neighbor with $k = 1, 3, 5$.
(4–8) *mlp*: Multilayer perceptron was with $D$ inputs and $K$ classes, the number of hidden units is taken as $D(ml1)$, $K(ml2)$, $(D+K)/2(ml3)$, $D+K(ml4)$, $2(D+K)(ml5)$.
(9) *lnp*: Linear perceptron with softmax outputs trained by gradient-descent to minimize cross-entropy.
(10) *c45*: The most widely used C4.5 decision tree algorithm.
(11) *mdt*: This is a multivariate decision tree where splits are hyperplanes that use all inputs as opposed to univariate trees, which use a single feature and implements an axis-orthogonal split [60].
(12–14) *svm*: Support vector machines (SVM) with a linear kernel (*sv1*), polynomial kernel of degree 2 (*sv2*), and a radial (Gaussian) kernel (*svr*). We use the LIBSVM 2.82 library that implements pairwise SVMs [11].

### 4.3. Division of training, validation, and test sets

Our methodology is as follows: a given data set is first divided into two parts, with 1/3 as the test set, *test*, and 2/3 as the training set, *train-all*. The training set, *train-all*, is then resampled using $5 \times 2$ cross-validation (cv) [14] where twofold cv is done five times (with stratification), and the roles swapped at each fold to generate 10 training and validation folds, $tra_i$, $val_i$, $i = 1, \ldots, 10$. $tra_i$ are used to train the base classifiers (discriminants). $val_i$ are divided into two randomly as $val - A_i$ and $val - B_i$, where $val - A_i$ are used to train the combiner and $val - B_i$ are used for model selection (in choosing the optimal subset or to choose the size of generated subsets). These ten trained models (base classifiers and combiner) are tested on the same *test* and we have ten $test_i$ accuracy results. These processed data of base classifier outputs are publicly available [61].

To compare the accuracies of different ensemble construction methods for statistically significant difference, we use two different methodologies. First, for each data set, we use the $5 \times 2$ cv $F$-test [3] ($\alpha = 0.05$), which is a parametric test to compare the methods for each data set; we then use the sign test to check if the numbers of wins/losses over all 38 data sets is significant. Second, we use Friedman's test which is a nonparametric test using the rankings, and if it rejects, we use the Nemenyi test as a post hoc test to check for significant difference between methods [13].

### 4.4. Compared ensembles

The following ensembles are generated and compared:

- Best: We order the base classifiers in terms of accuracy and use the first 1, 3, 5, 7, 9 of them. This has two variants: Best.sum uses the fixed sum rule and Best.lin uses the trained linear combiner.
- Rnd: We randomly choose 1, 3, 5, 7, 9 base classifiers, with Sum and Lin options.
- All: All the available base classifiers are combined without selection, with Sum and Lin options.
- Opt: We try all possible subsets (there are $2^{14}$) and choose the most accurate . It has Sum and Lin options.
- Icon: The classifier ensembles generated by Icon variants (Acc, Cv, Qstat, and Corr) are used. They all have Sum and Lin options.
- Fss: This is the discriminant ensemble, which uses forward subset selection and a linear combiner.
- Dt: This is the discriminant ensemble, which uses a decision tree.
- Dt.lin: This is the discriminant ensemble, which uses a decision tree for feature selection and a linear combiner.

**Table 3**
Accuracies of compared methods. ★ means Acc is significantly more accurate, ∘ means Acc is significantly less accurate.

| Data set | Acc | Dt | Dt.lin | Fss | Opt | Best | All | Rnd |
|---|---|---|---|---|---|---|---|---|
| Australian | 85.2 ± 0.6 | 78.2 ± 4.6 | 82.8 ± 0.5★ | 86.5 ± 0.6 | 85.2 ± 1.1 | 84.8 ± 0.5 | 85.0 ± 1.2 | 85.7 ± 1.3 |
| Balance | 92.3 ± 2.0 | 94.5 ± 2.6 | 91.7 ± 0.2 | 91.4 ± 0.2 | 92.3 ± 2.0 | 91.1 ± 1.7 | 90.0 ± 0.6 | 90.5 ± 0.8 |
| Breast | 94.4 ± 0.1 | 90.9 ± 6.7 | 93.6 ± 1.3 | 94.4 ± 0.1 | 94.4 ± 0.3 | 94.4 ± 0.3 | 94.6 ± 0.4 | 94.4 ± 0.3 |
| Bupa | 70.8 ± 3.2 | 57.7 ± 10.4 | 61.2 ± 0.0★ | 64.4 ± 5.4 | 69.8 ± 2.6 | 68.7 ± 2.9 | 66.1 ± 3.6 | 66.2 ± 2.5 |
| Car | 96.5 ± 0.9 | 96.0 ± 3.4 | 96.9 ± 1.7 | 92.2 ± 0.1★ | 96.5 ± 0.9 | 96.7 ± 0.9 | 93.4 ± 1.1★ | 94.8 ± 1.5 |
| Cmc | 51.3 ± 1.3 | 44.0 ± 2.0★ | 44.3 ± 1.1★ | 49.0 ± 1.1★ | 51.2 ± 0.9 | 52.1 ± 1.4 | 50.9 ± 1.6 | 51.5 ± 0.9 |
| Credit | 84.1 ± 1.6 | 76.1 ± 6.8 | 82.5 ± 0.6 | 85.6 ± 0.7 | 85.7 ± 0.8 | 83.6 ± 1.0 | 85.1 ± 1.2 | 85.5 ± 1.3 |
| Cylinder | 70.2 ± 4.3 | 73.7 ± 3.7 | 80.5 ± 1.7∘ | 77.1 ± 0.8 | 72.1 ± 2.6 | 71.7 ± 3.5 | 72.1 ± 2.6 | 72.6 ± 2.8 |
| Dermatology | 96.4 ± 0.6 | 95.3 ± 1.1 | 85.5 ± 5.4★ | 91.2 ± 0.0★ | 96.4 ± 0.6 | 96.0 ± 1.1 | 95.8 ± 1.1 | 96.4 ± 0.6 |
| *E. coli* | 85.0 ± 2.6 | 68.5 ± 10.6 | 73.0 ± 6.3 | 79.7 ± 2.0 | 83.1 ± 3.7 | 84.0 ± 2.6 | 83.6 ± 3.7 | 81.3 ± 3.8 |
| Flags | 59.6 ± 1.6 | 40.9 ± 10.2 | 43.7 ± 9.6★ | 47.6 ± 3.2★ | 59.6 ± 1.6 | 58.1 ± 2.6 | 56.7 ± 2.2★ | 56.4 ± 2.4★ |
| Flare | 88.1 ± 0.0 | 73.0 ± 24.0 | 88.2 ± 0.3 | 88.1 ± 0.0★ | 88.1 ± 0.0 | 88.1 ± 0.0 | 88.1 ± 0.0 | 88.1 ± 0.0 |
| Glass | 59.3 ± 3.6 | 49.6 ± 10.2 | 56.9 ± 3.5 | 51.6 ± 2.1★ | 55.5 ± 2.8 | 56.5 ± 2.6 | 54.1 ± 2.8 | 56.4 ± 2.3 |
| Haberman | 74.5 ± 1.1 | 55.9 ± 12.1★ | 71.0 ± 1.3★ | 73.5 ± 0.0 | 73.7 ± 1.5 | 73.7 ± 1.0 | 73.7 ± 1.1 | 73.6 ± 1.0 |
| Heart | 84.7 ± 1.1 | 69.2 ± 8.9 | 81.1 ± 3.9 | 87.2 ± 2.3 | 84.9 ± 1.8 | 84.7 ± 1.1 | 86.0 ± 2.0 | 85.0 ± 2.2 |
| Hepatitis | 81.0 ± 2.5 | 80.2 ± 2.2 | 79.2 ± 1.2 | 78.8 ± 0.0 | 81.5 ± 1.6 | 81.0 ± 2.5 | 81.2 ± 1.5 | 81.0 ± 2.5 |
| Horse | 85.4 ± 1.5 | 77.1 ± 12.1 | 85.2 ± 0.9 | 87.7 ± 1.1 | 84.4 ± 2.1 | 85.4 ± 1.5 | 85.6 ± 1.8 | 84.0 ± 1.9 |
| Ionosphere | 91.5 ± 1.1 | 89.5 ± 5.2 | 87.4 ± 3.3 | 90.7 ± 0.6 | 89.8 ± 1.9 | 89.8 ± 1.9 | 86.4 ± 2.4 | 86.9 ± 1.7★ |
| Iris | 96.7 ± 1.9 | 89.4 ± 7.8 | 82.2 ± 14.3 | 92.2 ± 0.0★ | 96.7 ± 2.5 | 96.7 ± 1.9 | 95.1 ± 1.9 | 94.3 ± 3.1 |
| Monks | 89.7 ± 7.3 | 94.4 ± 5.5 | 96.0 ± 3.5 | 97.7 ± 1.3 | 89.7 ± 7.3 | 83.6 ± 8.9 | 76.7 ± 2.9 | 82.4 ± 6.6★ |
| Mushroom | 100.0 ± 0.1 | 100.0 ± 0.0 | 100.0 ± 0.0 | 100.0 ± 0.0 | 100.0 ± 0.1 | 100.0 ± 0.1 | 99.9 ± 0.1 | 99.9 ± 0.1 |
| Nursery | 99.8 ± 0.1 | 99.9 ± 0.1 | 98.9 ± 1.1 | 100.0 ± 0.0 | 99.8 ± 0.1 | 99.8 ± 0.2 | 99.2 ± 0.2★ | 99.3 ± 0.3★ |
| Optdigits | 98.2 ± 0.2 | 97.7 ± 0.4 | 98.2 ± 0.3 | 98.8 ± 0.1∘ | 98.2 ± 0.2 | 98.2 ± 0.2 | 97.8 ± 0.2 | 97.6 ± 0.2 |
| Pageblock | 96.2 ± 0.3 | 93.7 ± 1.4 | 96.4 ± 0.2 | 96.3 ± 0.2 | 96.1 ± 0.3 | 96.2 ± 0.2 | 95.8 ± 0.3 | 95.6 ± 0.3 |
| Pendigits | 99.2 ± 0.1 | 98.5 ± 0.9 | 99.2 ± 0.2 | 99.4 ± 0.0 | 99.1 ± 0.1 | 99.2 ± 0.1 | 99.0 ± 0.1★ | 98.9 ± 0.1★ |
| Pima | 74.0 ± 1.0 | 68.2 ± 2.3★ | 70.7 ± 0.9★ | 74.7 ± 0.3 | 74.8 ± 1.0 | 74.8 ± 0.9 | 74.2 ± 1.3 | 74.2 ± 0.9 |
| Ringnorm | 98.4 ± 0.1 | 98.0 ± 0.2 | 98.5 ± 0.1 | 98.5 ± 0.0 | 98.4 ± 0.1 | 98.4 ± 0.1 | 93.2 ± 0.4★ | 92.0 ± 0.8★ |
| Segment | 95.0 ± 0.4 | 86.1 ± 8.4 | 96.4 ± 0.5 | 96.1 ± 0.1∘ | 94.8 ± 0.4 | 94.8 ± 0.7 | 93.9 ± 0.6 | 95.2 ± 0.7 |
| Spambase | 93.7 ± 0.4 | 89.2 ± 3.7 | 93.7 ± 0.4 | 94.0 ± 0.2 | 93.7 ± 0.3 | 93.1 ± 0.6★ | 93.3 ± 0.4 | 93.4 ± 0.4 |
| Tae | 48.5 ± 7.7 | 46.7 ± 12.8 | 53.5 ± 8.7 | 44.4 ± 6.0 | 47.3 ± 7.1 | 48.5 ± 7.7 | 50.6 ± 7.3 | 42.7 ± 6.0 |
| Thyroid | 98.1 ± 0.2 | 97.1 ± 1.4 | 97.8 ± 0.4 | 97.1 ± 0.0★ | 98.1 ± 0.2 | 98.3 ± 0.2 | 98.0 ± 0.2 | 98.1 ± 0.2 |
| Tictactoe | 99.4 ± 0.0 | 96.7 ± 4.3★ | 99.1 ± 0.8 | 99.4 ± 0.0 | 99.2 ± 0.2 | 99.3 ± 0.1 | 98.9 ± 0.6★ | 99.3 ± 0.3 |
| Titanic | 80.7 ± 0.1 | 80.7 ± 0.0 | 80.6 ± 0.3 | 80.7 ± 0.0 | 80.7 ± 0.1 | 80.2 ± 0.6 | 79.0 ± 2.2 | 80.2 ± 0.1★ |
| Twonorm | 97.4 ± 0.1 | 95.5 ± 1.2 | 96.9 ± 0.1★ | 97.5 ± 0.0 | 97.3 ± 0.1 | 97.4 ± 0.1 | 97.4 ± 0.1 | 97.3 ± 0.1 |
| Vote | 94.4 ± 0.9 | 93.8 ± 2.1 | 94.0 ± 2.3 | 96.8 ± 0.7∘ | 94.9 ± 1.2 | 94.4 ± 0.9 | 94.8 ± 0.8 | 96.4 ± 0.3∘ |
| Wine | 99.2 ± 1.4 | 93.7 ± 6.1 | 86.5 ± 10.6 | 96.5 ± 1.7 | 98.0 ± 1.1 | 97.3 ± 1.4 | 98.3 ± 1.1 | 98.3 ± 1.1 |
| Yeast | 58.4 ± 1.1 | 40.3 ± 3.0★ | 51.1 ± 0.3★ | 53.7 ± 0.6★ | 59.7 ± 1.3 | 59.0 ± 0.5 | 60.0 ± 1.3 | 60.3 ± 1.2 |
| Zoo | 94.1 ± 3.8 | 59.2 ± 7.3★ | 56.8 ± 0.0★ | 56.8 ± 0.0★ | 94.1 ± 3.8 | 94.1 ± 3.8 | 87.8 ± 3.9 | 96.2 ± 2.3 |

Accuracies of these algorithms on each data set can be seen in Table 3. The entries marked with ★ means that Acc is significantly more accurate than that algorithm using $5 \times 2$ *F*-test, ∘ means Acc is significantly less accurate.

## 5. Experimental results

First, we check for the effect of search direction in Icon. In the next subsection, we discuss and compare the results of our proposed methods on two data sets in detail, before moving on to an overall comparison on all data sets.

### 5.1. Comparison of search methods

As we have mentioned in Section 1, it is possible to search in the forward direction adding one at a time, backward direction removing one at a time, or a floating search where we try to remove a previously added base classifier before adding another one. We can hence consider three Icon variants implementing forward (.f), backward (.b) or floating (.l) search. The comparison of the accuracies of the three search directions according to the four criteria used by Icon variants can be seen in Table 4. These entries are the number of data sets, on which there is a statistically significant win/loss of the method in the row over the method in the column (using $5 \times 2$ cv *F*-test); 38 wins–losses give the number of ties; the entry is bold if the number of wins/losses is significant in 38 trials (using the sign test). The average and standard deviation of the number of base classifiers in the found ensemble and the number of visited states during search are given, respectively, in Tables 5 and 6.

We see that in all four criteria of Acc, Cv, Qstat, and Corr, .f and .l give similar results; this is because most of the time small ensembles are enough, and there is not much to prune back after few additions. We also see that in terms of the ensemble sizes and search time, .l and .f stand out as the best. With the diversity measure Corr, backward search is significantly more accurate and faster because Corr needs larger ensembles, and that is why with the diversity-based measures, forward and floating search take more steps.

**Table 4**
The number of statistically significant accuracy wins/losses of .F (Forward), .B (Backward), and .L (Floating) over 38 data sets according to the criterion used by Icon. The bold face entries show statistically significantly difference using sign test.

| Acc | .F | .B | .L | Cv | .F | .B | .L |
|---|---|---|---|---|---|---|---|
| .F | 0/0 | 1/0 | 1/0 | .F | 0/0 | 1/5 | 0/0 |
| .B | 0/1 | 0/0 | 0/1 | .B | 5/1 | 0/0 | 5/1 |
| .L | 0/1 | 1/0 | 0/0 | .L | 0/0 | 1/5 | 0/0 |
| Qstat | .F | .B | .L | Corr | .F | .B | .L |
| .F | 0/0 | 2/5 | 2/0 | .F | 0/0 | **1/8** | 4/0 |
| .B | 5/2 | 0/0 | 8/2 | .B | **8/1** | 0/0 | **11/1** |
| .L | 0/2 | 2/8 | 0/0 | .L | 0/4 | **1/11** | 0/0 |

**Table 5**
Average ± standard deviation of the number of classifiers in ensembles found by each search direction and optimization criterion.

| | Acc | Cv | Qstat | Corr |
|---|---|---|---|---|
| .F | 2.39 ± 1.5 | 1.05 ± 0.2 | 8.16 ± 4.9 | 5.74 ± 5.1 |
| .B | 7.79 ± 3.2 | 2.08 ± 2.1 | 11.55 ± 3.7 | 11.39 ± 3.7 |
| .L | 2.26 ± 1.3 | 1.05 ± 0.2 | 6.29 ± 4.4 | 3.32 ± 2.3 |

**Table 6**
Average ± standard deviation of the number of search steps visited by each search direction and optimization criterion.

| | Acc | Cv | Qstat | Corr |
|---|---|---|---|---|
| .F | 42.37 ± 15.1 | 27.63 ± 2.7 | 143.05 ± 28.6 | 125.84 ± 30.6 |
| .B | 73.47 ± 22.9 | 101.68 ± 8.0 | 37.26 ± 29.0 | 38.95 ± 29.0 |
| .L | 712.47 ± 345.4 | 415.58 ± 40.6 | 172.13 ± 61.2 | 131.68 ± 40.6 |

**Table 7**
Single classifier accuracies on *optdigits*.

| Alg | *test* |
|---|---|
| *c45* | 81.76 ± 1.3 |
| *mdt* | 92.98 ± 1.0 |
| *lnp* | 94.28 ± 0.8 |
| *ml2* | 94.86 ± 0.5 |
| *ml4* | 96.03 ± 0.9 |
| *ml5* | 96.21 ± 0.4 |
| *ml1* | 96.02 ± 0.6 |
| *3nn* | 95.99 ± 0.4 |
| *1nn* | 96.67 ± 0.4 |
| *5nn* | 95.86 ± 0.3 |
| *ml3* | 96.34 ± 0.3 |
| *sv1* | 97.48 ± 0.2 |
| *svr* | 97.67 ± 0.2 |
| *sv2* | 97.49 ± 0.3 |

We can see that .B and .L increase both the number of search steps and the ensemble size for Acc and Cv. We believe that it is not beneficial to use floating search because it finds the same ensembles as forward search does, but takes more steps. In the diversity-based measures, backward search is more accurate but this increases the ensemble size and we have no benefit over using the whole ensemble. Aiming high accuracy and small ensembles, we therefore adopt forward search in the rest of the paper.

### 5.2. Initial results

We start by discussing in detail our results on two data sets, *optdigits* and *nursery*, as two example cases. We present our overall results on all data sets in the next subsection.

**Table 8**
Combination results on *optdigits*.

| Alg | *test* | # cla | # disc | Chosen |
|---|---|---|---|---|
| Best.1.sum | 97.49 ± 0.3 | 1 | 10 | *sv2* |
| Best.3.sum | 98.22 ± 0.2 | 3 | 30 | *sv2 svr sv1* |
| Best.5.sum | 98.19 ± 0.2 | 5 | 50 | *sv2 svr sv1 ml3 5nn* |
| Best.7.sum | 97.90 ± 0.2 | 7 | 70 | *sv2 svr sv1 ml3 5nn 1nn 3nn* |
| Best.9.sum | 98.01 ± 0.3 | 9 | 90 | *sv2 svr sv1 ml3 5nn 1nn 3nn ml1ml5* |
| Rnd.1.sum | 97.67 ± 0.2 | 1 | 10 | *svr* |
| Rnd.3.sum | 97.60 ± 0.3 | 3 | 30 | *c45 sv1 1nn* |
| Rnd.5.sum | 97.67 ± 0.2 | 5 | 50 | *ml3 ml1 ml5 sv2 1nn* |
| Rnd.7.sum | 97.82 ± 0.2 | 7 | 70 | *ml3 ml4 mdt ml5 svr sv1 1nn* |
| Rnd.9.sum | 97.61 ± 0.2 | 9 | 90 | *ml2 c45 ml3 lnp mdt ml5 svr sv2 3nn* |
| All.sum | 97.85 ± 0.2 | 14 | 140 | |
| Opt.sum | 98.22 ± 0.2 | 3 | 30 | *svr sv2 sv1* |
| Acc.sum | 98.22 ± 0.2 | 3 | 30 | *sv2 svr sv1* |
| Cv.sum | 96.02 ± 0.6 | 1 | 10 | *ml1* |
| Qstat.sum | 97.85 ± 0.2 | 14 | 140 | All |
| Corr.sum | 97.85 ± 0.2 | 14 | 140 | All |
| Fss | 98.83 ± 0.1 | 10 | 25 | *c45*(1,4) *1nn*(3) *3nn*(3,5) *5nn*(7) *ml1*(9) *ml2*(4) *ml3*(1) *sv1*(2,3,5,6,7) *sv2*(1,2,4,7,8,9) *svr*(0,3,4,6,8) |
| Dt | 97.65 ± 0.4 | 6.6 | 11.8 | *1nn*(6) *5nn*(5) *lnp*(0) *sv1*(7) *sv2*(1,2,3,4) *svr*(8,9) |
| Dt.lin | 98.22 ± 0.3 | 6.6 | 11.8 | *1nn*(6) *5nn*(5) *lnp*(0) *sv1*(7) *sv2*(1,2,3,4) *svr*(8,9) |

### 5.2.1. Optdigits data set

The accuracies of base classifiers (sorted in increasing accuracy) are shown in Table 7 and the ensembles on *test* data are given in Table 8, together with the number of classifiers, the number of discriminants, and the chosen ensembles. The plot of accuracies vs. the number of base classifiers is shown in Fig. 2. On this data set (as many others), using a linear combiner .Lin does not increase the accuracy significantly, and therefore only .Sum results are given to keep the table and figures simpler.

We see that the optimal subset, Opt, chooses only 3 of 14 and is significantly as accurate as All that uses all 14. Ordering base classifiers in terms of accuracy and choosing the best $m$, Best, are also significantly better than All. Note that the accuracy of Best starts to decrease when too many base classifiers are added; in such a case the increase in bias is higher than the decrease in variance. Choosing a random subset, Rnd, does not work as well and requires more base classifiers.

In terms of the ensembles of classifiers generated by Icon, we see that Acc also finds the optimal subset. The optimal subset generally contains a few base classifiers and searching in the forward direction, as Icon variant Acc does, greedily returns a quite good subset in polynomial time.

The Icon variants that use diversity measures, Qstat and Corr, use too many base classifiers, showing that it is best to use accuracy as the ensemble evaluation criterion directly rather than an intermediate diversity measure; or that diversity should not be used alone but in some combination with accuracy [28,29].

Cv uses only one base classifier as it finds that adding the second does not increase accuracy significantly. The behavior of Cv depends on the statistical test and the confidence level. We show in Fig. 3 how the ensemble changes at different confidence levels. When the confidence level is lower than 0.95, the test uses a smaller confidence interval, which is more likely to

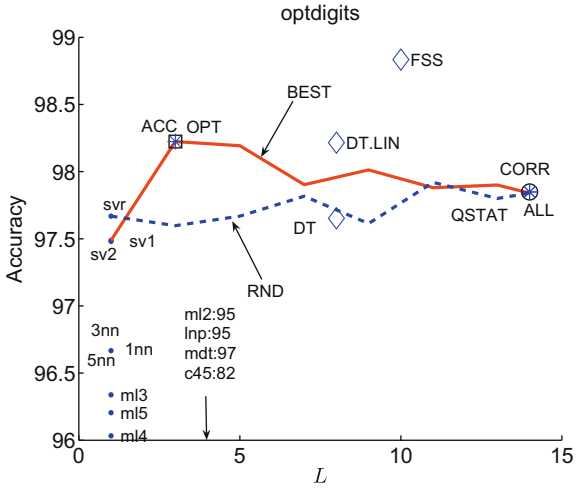


**Fig. 2.** Accuracy vs. the number of classifiers on *optdigits test*.

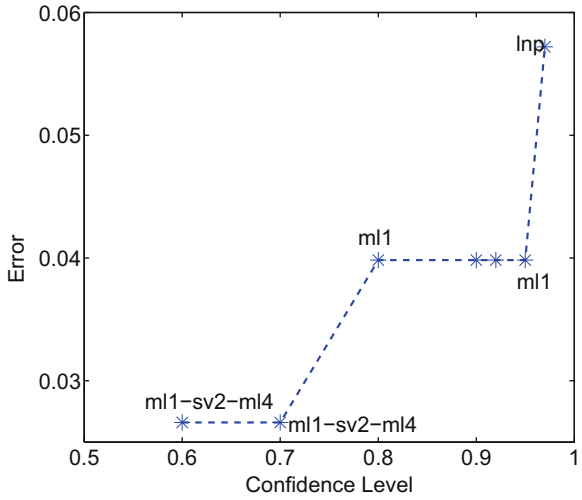


**Fig. 3.** The ensemble found by Cv changes as the confidence level changes on *optdigits*.

reject, and generates larger ensembles: Cv tends to behave similar to Acc as the confidence level decreases. As the confidence level increases, the test gets more and more conservative and generates smaller ensembles, until it chooses only one.

As for the ensembles of discriminants, we see that Fss uses more discriminants than Dt, but is significantly more accurate than all other methods. On the average, Dt chooses 11.8 discriminants (the smallest possible with 10 classes is 9) from 6.6 classifiers. Example tree (one of 10) learned by Dt is given in Fig. 4, and its discriminants are given in Table 8: it starts by looking at the output of *sv1* for class '7' and chooses '7' if this value is higher than 0.46. Note that we only evaluate the discriminants in our path; for example, we see here that the complex *5nn* is only evaluated to distinguish '0', '1' and '6' from others. Dt.lin improves over Dt, but not significantly ($p = 0.90$). Most classes are identifiable by looking at a single discriminant; only '1' requires two (*sv2* and *1nn*). The example tree Dt chooses only 10 discriminants, choosing discriminants from six base classifiers; this set includes the optimal subset found by Opt (which has $3 \times 10 = 30$ discriminants), and uses three more discriminants from three classifiers (*5nn*, *1nn*, and *lnp*).

### 5.2.2. Nursery data set

The results on *nursery* is similar except that .Lin improves accuracy over .Sum, for some ensemble methods. The results for the single base classifiers are given in Table 9 and the combination methods are given in Table 10. In Fig. 5, we compare the .Sum and .Lin variants.

On this data set, the accuracies of the base classifiers range from 76.85 to 99.41 and we see that using .Sum (Fig. 5a), All has low accuracy and the accuracy of Best decreases as more classifiers are added; we do not see this as strongly with .Lin

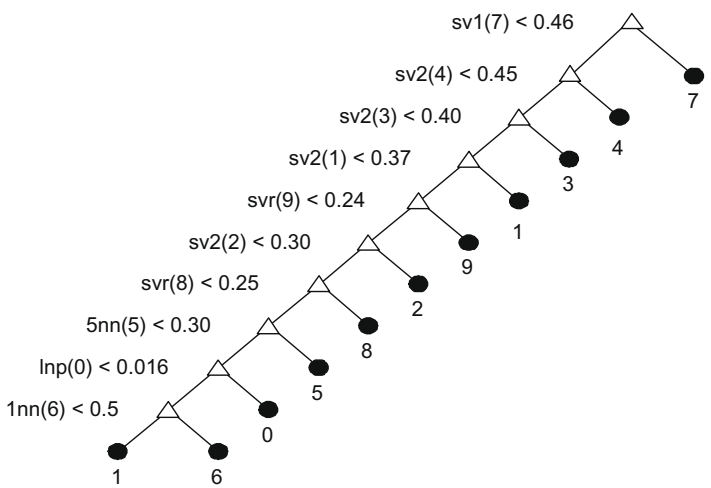


**Fig. 4.** One of the decision trees learned by Dt on *optdigits*.

**Table 9**
Single classifier accuracies on *nursery* data set.

| Alg | *test* |
|---|---|
| *1nn* | 76.86 ± 0.5 |
| *3nn* | 85.18 ± 0.5 |
| *5nn* | 89.93 ± 0.4 |
| *lnp* | 90.86 ± 0.7 |
| *sv1* | 92.43 ± 0.3 |
| *mdt* | 92.74 ± 0.4 |
| *c45* | 92.72 ± 0.5 |
| *ml2* | 94.97 ± 0.9 |
| *svr* | 95.50 ± 0.4 |
| *sv2* | 98.70 ± 0.6 |
| *ml3* | 99.13 ± 0.4 |
| *ml5* | 99.38 ± 0.3 |
| *ml4* | 99.42 ± 0.3 |
| *ml1* | 99.41 ± 0.2 |

**Table 10**
Combination results on *nursery* data set.

| Alg | *test* | # cla | # disc | Chosen |
|---|---|---|---|---|
| BEST.1.LIN | 99.41 ± 0.2 | 1 | 4 | *ml1* |
| BEST.3.LIN | 99.58 ± 0.2 | 3 | 12 | *ml1 ml4 ml3* |
| BEST.5.LIN | 99.81 ± 0.1 | 5 | 20 | *ml1 ml4 ml3 ml5 sv2* |
| BEST.7.LIN | 99.78 ± 0.1 | 7 | 28 | *ml1 ml4 ml3 ml5 sv2 svr ml2* |
| BEST.9.LIN | 99.77 ± 0.1 | 9 | 36 | *ml1 ml4 ml3 ml5 sv2 svr ml2 c45 mdt* |
| RND.1.LIN | 91.48 ± 0.3 | 1 | 4 | *5nn* |
| RND.3.LIN | 99.40 ± 0.2 | 3 | 12 | *ml1 3nn svr* |
| RND.5.LIN | 99.51 ± 0.3 | 5 | 20 | *lnp ml4 ml5 1nn svr* |
| RND.7.LIN | 99.60 ± 0.2 | 7 | 28 | *ml2 ml3 ml1 ml4 mdt ml5 1nn* |
| RND.9.LIN | 99.75 ± 0.1 | 9 | 36 | *ml2 ml3 lnp ml1 ml4 c45 5nn sv1 sv2* |
| ALL.LIN | 99.76 ± 0.1 | 14 | 56 | |
| OPT.LIN | 99.81 ± 0.1 | 5 | 20 | *ml3 ml1 ml5 5nn sv2* |
| ACC.LIN | 99.82 ± 0.1 | 4 | 16 | *ml1 sv2 ml4 3nn* |
| CV.LIN | 99.83 ± 0.1 | 2 | 8 | *ml1 sv2* |
| QSTAT.LIN | 93.13 ± 0.5 | 2 | 8 | *c45 1nn* |
| CORR.LIN | 98.91 ± 0.4 | 2 | 8 | *ml3 1nn* |
| FSS | 99.95 ± 0.0 | 5 | 8 | *ml1*(0) *ml3*(0) *ml4*(0,3) *mdt*(0) *sv2*(0,2,3) |
| DT | 99.95 ± 0.1 | 2.7 | 3 | *c45*(1) *sv2*(2,3) |
| DT.LIN | 98.85 ± 1.1 | 2 | 3 | *c45*(1) *sv2*(2,3) |

(Fig. 5b). That is, a trained combiner is more robust to the addition of erroneous classifiers. The trained combiner is able to weight the accurate ones more and effectively ignores those that are not as accurate. We also see that ICON, as a subset selection method, shows the same robust behavior; that is, because it does not add and use the inaccurate classifiers, its accuracy is not degraded. Note that this is valid even if .SUM is used (Fig. 5a). We therefore see that in the presence of inaccurate base classifiers in the ensemble, either one should use a trained combiner, which learns to ignore them or use a subset selection method that does not include them.

This is a data set with four classes and the tree learned by DT for this fold (Fig. 6) has only three decision nodes, is very simple, and has 100.00 *test* accuracy (accuracy of DT.LIN is 99.31), which is higher than what we get when we use all outputs of all base classifiers for that fold (99.81). FSS uses more discriminants and is accurate. On this data set, ACC, OPT, FSS, and BEST are significantly more accurate than ALL and RND.

### 5.3. General results

We compare the accuracies of all ensemble methods in a pairwise manner on *test* in Table 11. These are the number of significant wins and losses of method in the row over the method in the column. The sum of wins and losses subtracted from 38 gives the number of ties. The entry in bold indicates that the number of wins/losses over 38 is statistically significant using the sign test. We do further statistical analysis with nonparametric tests using the average ranks of the six ensemble methods on 38 data sets (Table 12). Friedman's test rejects the hypothesis that the six methods have equal ranks. Doing Nemenyi's post hoc test for pairwise comparison, we get the results in Fig. 7.

Table 13 shows the average number of base classifiers (/discriminants) contained in the ensembles that are constructed using different methods. The discriminant values are normalized by dividing with the number of classes.

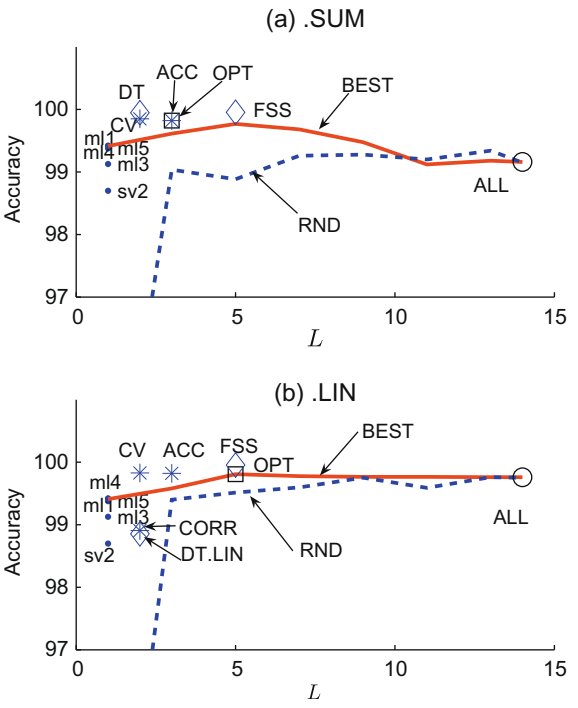


**Fig. 5.** Comparison of fixed .Sum and trained .Lin on *nursery test*.

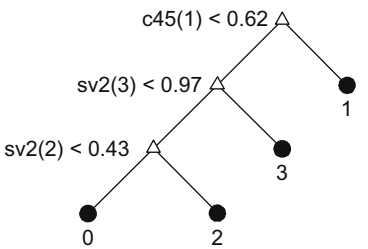


**Fig. 6.** One of the decision trees learned by Dt on *nursery*.

Additional to their accuracies and complexities, we also check how similar are the ensembles found by different methods. Given two ensembles $E_i$ and $E_j$, we define the similarity between them as the number of shared base classifiers (or discriminants):

$$\mathrm{Sim}(E_i, E_j) = \frac{|E_i \cap E_j|}{|E_i \cup E_j|} \tag{5}$$

If the two ensembles share the same base classifiers (discriminants), the similarity is 1; if there is no intersection, the similarity is 0. The average similarity between ensembles found by different methods is given in Table 14.

Analyzing these tables, our general results are as follows:

- Overall, using a trained linear combiner does not increase the ensemble accuracy significantly. In Table 15, we report a comparison of accuracies using the fixed sum rule .Sum vs. the trained linear combiner .Lin; we see that except with Cv, there is no significant difference between them, and on Cv, it is .Sum which is more accurate.
  We believe this is because we have sufficiently well-trained, capable base classifiers, which are able to approximate posterior probabilities quite well that no further bias correction (which the linear combiner effectively is there for) is required. It may also be the case that on small data sets, there is not enough data to train the linear combiner sufficiently. It is possible to increase the number of folds and, for example, use $20 \times 1$ cross-validation that will have the effect of increasing the training set both for base classifiers and the combiner at the expense of doubling the running time. In

**Table 11**
Pairwise comparison of accuracies (wins/losses over 38) of all methods using $5 \times 2$ cv $F$-test.

| | Best | Rnd | All | Opt | Acc | Cv | Qstat | Corr | Fss | Dt | Dt.Lin |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Best | 0/0 | 3/1 | **6/0** | 0/0 | 0/1 | **9/1** | **7/0** | **8/0** | 10/4 | 6/1 | **9/1** |
| Rnd | 1/3 | 0/0 | 4/2 | **0/7** | 1/7 | **9/1** | 8/2 | **10/1** | 7/6 | 7/3 | 10/4 |
| All | **0/6** | 2/4 | 0/0 | **0/6** | **0/6** | 9/3 | 5/2 | 8/2 | 6/9 | 6/3 | 8/6 |
| Opt | 0/0 | **7/0** | **6/0** | 0/0 | 0/0 | **12/0** | **9/0** | **9/0** | 9/6 | **6/0** | 9/2 |
| Acc | 1/0 | 7/1 | **6/0** | 0/0 | 0/0 | **14/2** | **10/1** | **9/1** | 10/3 | **6/0** | **10/1** |
| Cv | **1/9** | **1/9** | 3/9 | **0/12** | **2/14** | 0/0 | 6/7 | 6/7 | 3/9 | 4/1 | 5/6 |
| Qstat | **0/7** | 2/8 | 2/5 | **0/9** | **1/10** | 7/6 | 0/0 | 5/1 | 5/12 | 5/5 | 8/8 |
| Corr | **0/8** | **1/10** | 2/8 | **0/9** | **1/9** | 7/6 | 1/5 | 0/0 | 7/12 | 5/5 | 5/8 |
| Fss | 4/10 | 6/7 | 9/6 | 6/9 | 3/10 | 9/3 | 12/5 | 12/7 | 0/0 | 7/1 | **12/3** |
| Dt | 1/6 | 3/7 | 3/6 | **0/6** | **0/6** | 1/4 | 5/5 | 5/5 | 1/7 | 0/0 | 1/4 |
| Dt.lin | **1/9** | 4/10 | 6/8 | 2/9 | **1/10** | 6/5 | 8/8 | 8/5 | **3/12** | 4/1 | 0/0 |

**Table 12**
Average ranks of compared methods.

| Best | Rnd | All | Opt | Acc | Cv | Qstat | Corr | Fss | Dt | Dt.Lin |
|---|---|---|---|---|---|---|---|---|---|---|
| 4.55 | 5.68 | 5.76 | 3.89 | 4.08 | 8.01 | 6.97 | 7.17 | 4.82 | 8.62 | 6.43 |

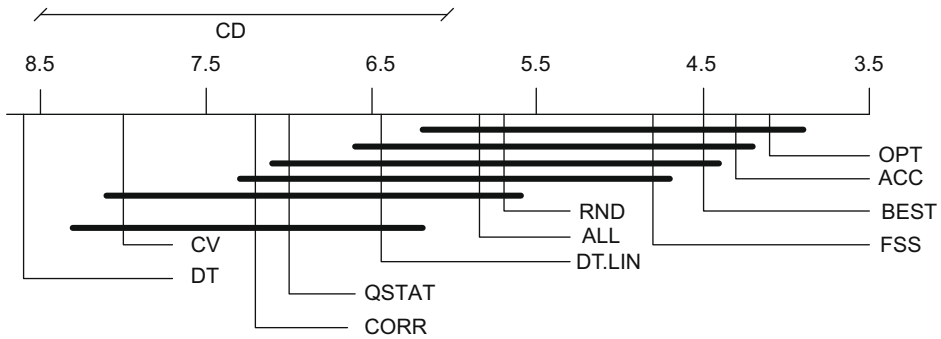


**Fig. 7.** Graphical representation of post hoc Nemenyi test results of compared methods with ranks given in Table 12 [13]. The numbers on the line represent the average ranks, CD is the critical difference for statistical significance, and bold lines connect the methods that have no significant difference.

**Table 13**
Average number of base classifiers (/discriminants) contained in different ensembles.

| | Best | Rnd | All | Opt | Acc | Cv | Qstat | Corr | Fss | Dt |
|---|---|---|---|---|---|---|---|---|---|---|
| Sum | 2.79 | 4.89 | 14.00 | 4.53 | 2.39 | 1.05 | 8.16 | 5.74 | 4.58/1.83 | 5.60/2.76 |
| Lin | 5.84 | 6.58 | 14.00 | 6.08 | 3.95 | 1.08 | 8.16 | 5.74 | | 5.60/2.76 |

the extreme case, one can use leave-one-out where the number of folds is equal to the training set size (as proposed in the original stacking paper of Wolpert [58]) but this cannot be afforded (nor is it necessary) unless the data set is small.Using a fixed rule decreases time/space complexity of training/testing, does not require data to train the combiner allowing more data to train base classifiers, and is simpler to interpret. Therefore throughout the rest of this section, we consider .Sum results only.

- According to Nemenyi's test given in Fig. 7, there is no significant difference in accuracy between Best, Opt, Rnd, Acc, Fss. Among those, as we see in Table 13, Acc uses the least number of classifiers, and Fss uses the least number of discriminants. These results show that our proposed methods for the ensembles of classifiers and discriminants construct ensembles which are simple and as accurate as optimal subset or the whole.
- Among Icon variants, Acc is the most accurate. It is also more accurate than All on 6 data sets of 38 on *test* (which is significant using the sign test). Acc does not loose to Opt on any data set. The similarity of the ensembles they find is 0.48; this is the fourth highest value in the table—the highest similarity is between Qstat and Corr with 0.66. This shows that a greedy, forward, incremental accuracy-based ensemble construction method works quite well. Cv rarely uses more than a single base classifier, and the diversity-based Icon variants are significantly less accurate than Acc.

**Table 14**
Average similarity of base classifiers (discriminants) between ensembles found by different methods.

| | Best | All | Opt | Acc | Cv | Qstat | Corr | Fss | Dt |
|---|---|---|---|---|---|---|---|---|---|
| Best | 1.00 | 0.20 | 0.44 | 0.61 | 0.11 | 0.13 | 0.14 | 0.32<br>0.20 | 0.20<br>0.13 |
| All | 0.20 | 1.00 | 0.32 | 0.17 | 0.08 | 0.58 | 0.41 | 0.33<br>0.13 | 0.40<br>0.20 |
| Opt | 0.44 | 0.32 | 1.00 | 0.48 | 0.14 | 0.23 | 0.20 | 0.37<br>0.21 | 0.28<br>0.17 |
| Acc | 0.61 | 0.17 | 0.48 | 1.00 | 0.09 | 0.13 | 0.15 | 0.29<br>0.19 | 0.19<br>0.14 |
| Cv | 0.11 | 0.08 | 0.14 | 0.09 | 1.00 | 0.06 | 0.09 | 0.16<br>0.12 | 0.07<br>0.05 |
| Qstat | 0.13 | 0.58 | 0.23 | 0.13 | 0.06 | 1.00 | 0.66 | 0.23<br>0.11 | 0.32<br>0.17 |
| Corr | 0.14 | 0.41 | 0.20 | 0.15 | 0.09 | 0.66 | 1.00 | 0.24<br>0.14 | 0.24<br>0.14 |
| Fss | 0.32<br>0.20 | 0.33<br>0.13 | 0.37<br>0.21 | 0.29<br>0.19 | 0.16<br>0.12 | 0.23<br>0.11 | 0.24<br>0.14 | 1.00<br>1.00 | 0.26<br>0.12 |
| Dt | 0.20<br>0.13 | 0.40<br>0.20 | 0.28<br>0.17 | 0.19<br>0.14 | 0.07<br>0.05 | 0.32<br>0.17 | 0.24<br>0.14 | 0.26<br>0.12 | 1.00<br>1.00 |

**Table 15**
Comparison of accuracies (wins/losses over 38) of .Sum vs .Lin.

| | Best | Rnd | All | Opt | Acc | Cv | Qstat | Corr |
|---|---|---|---|---|---|---|---|---|
| *test* | 2/1 | 2/3 | 0/5 | 0/0 | 1/0 | **6/0** | 3/6 | 5/5 |

- Best is also as accurate as Opt but needs more base classifiers than Acc does. The similarity between Acc and Best ensembles is 0.61 (which is the second highest value in the table), showing that the greedy algorithm selects from the best $m$ but sometimes it chooses differently. Acc ensembles contain fewer base classifiers than those of Best, indicating that instead of choosing the best $m$ classifiers it is better to do some choosing for those which complement each other and then we can do with less.
- Even Rnd works quite well but needs a larger ensemble, requiring more than two times more base classifiers than Acc does.
- According to the average number of base classifiers they choose to include in their ensembles, we have the following ordering: Cv < Acc < Best < Opt < Fss < Rnd < Dt < Corr < Qstat < All.
- Acc and Cv use different base classifiers, because Acc selects the classifier with the highest accuracy, whereas Cv also takes complexity into account and chooses the classifier which is the least complex among the most accurate. Because Cv ensembles contain a single base classifier, they are generally less accurate than other ensembles.
- In 32 of the 38 data sets, Opt contains Best.1, which means that an incremental algorithm beginning with the best individual algorithm has higher probability of finding Opt than an incremental algorithm beginning with a random classifier. On the remaining six data sets, Opt contains the second best algorithm (and on five of them Opt contains the same algorithm as Best.1 with a different hyperparameter.).
- Cv requires statistically significant difference for a classifier that is to be added and rarely adds more than one classifier (average number of base classifiers found by this method is 1.05), but Acc, which looks at accuracy only, may achieve statistically significant improvement after more than one step of classifier addition.
- Diversity-based methods, Qstat and Corr, do not work well. Most of the other methods are significantly more accurate than them. The similarity between the ensembles found by these two is 0.66, indicating that they tend to find similar (but not very similar) ensembles, but their similarity to Opt is around 0.20–0.25, showing that they do not choose good base classifiers. Their similarity to other accuracy-based methods (other than all) is also around 0.2. These results indicate that rather than using such diversity measures alone, it is best to combine them with accuracy in some general goodness measure.
- Among the three discriminant ensembles, Fss is significantly more accurate than Dt.lin and more accurate than Dt (with seven wins to one losses; this is significant at $p = 0.93$). Fss on the average chooses fewer discriminants than Dt variants, and the similarity of discriminants selected by them is only 0.26. We believe this is because Fss chooses a feature that decreases the error over the whole training set, while Dt tries to minimize error over the data arriving to the current node (subtree) by choosing a different feature, and when we take a union over such features we get a larger set.

- DT.LIN seems to be more accurate than DT (with four wins to one loss, but it is not significant: $p = 0.62$) and has less variance, but using an additional linear combiner has disadvantages that we cited above. DT also has the advantages of being a tree: it is interpretable and during test, we only need to evaluate the nodes (discriminants) in our path, implying faster test performance.
- All discriminant ensembles are as accurate as ALL (FSS has nine wins, six losses compared to ALL). This implies that a discriminant ensemble is simple, interpretable and accurate.
- ACC uses fewer classifiers but FSS uses fewer discriminants. ACC uses all the discriminants of the chosen classifiers, which makes it costly but robust to noise due to redundancy; FSS and DT use a subset of the discriminants, are less noise-tolerant, but are simpler and faster.
- The similarity between the classifiers used by FSS and ACC is 0.29, whereas the similarity between ACC and DT is 0.19. Both being incremental forward methods, ACC and FSS tend to choose more similar classifiers, slightly different from those selected by a decision tree.
- Although the classifier and discriminant ensembles seem to contain a comparable number of base classifiers, in terms of the discriminants, the discriminant ensembles tend to need around half as many base discriminants, cutting by half the space and time complexity of training/testing. On some data sets, this can be as low as one-fifth.

### 5.4. Comparison with ensemble techniques

We also compared our proposed algorithms with the most frequently used ensemble techniques such as AdaBoost [16], bagging [8] and a variant of the random subspace [22] ensemble proposed by Bertoni et al. [7]. Theory of bagging has been analyzed by Fumera et al. [17]. The accuracy of these ensemble methods can be increased by combining with other methods (Zhang and Zhang [62] combined AdaBoost with rotation forests to come up with a new ensemble technique called RotBoost) or changing the order of aggregation [38]. We compare ACC and DT.LIN with the following algorithms:

- ADA: Standard AdaBoost implemented as proposed in [16]. We make a slight change and instead of stopping at 100% accuracy, we reset the instance weights and continue. We train decision tree ensembles of size 5, 10, 15, 20, 25, 30 and choose the one with the best *val-B* accuracy.
- BAG: The original bagging algorithm proposed in [8]. We train decision tree ensembles of size 5, 10, 15, 20, 25, 30 and choose the one with the best *val-B* accuracy.
- KNE: We train $k$-nn ensembles ($k = 1, 3, 5, 7, 9$) using random subspace method (we choose a different set of attributes for each fold) and combine them using voting.

We compare accuracies of these methods in a pairwise manner on *test* in Table 16. These are the number of significant wins and losses of method in the row over the method in the column. The sum of wins and losses subtracted from 38 gives the number of ties. If the entry is bold, this means that the number of wins/losses over 38 is statistically significant using the sign test. We do further statistical analysis with nonparametric tests using the average ranks of the five ensemble methods on 38 data sets (Table 18). Friedman's test rejects the hypothesis that the five methods have equal ranks. Doing Nemenyi's post hoc test for pairwise comparison, we get the results in Fig. 8. Looking at these tables, we can see by using the parametric sign test that ACC is better than all the other ensemble methods, there is no statistically significant difference between ADA and DT.LIN, and they are significantly more accurate than BAG and KNE. If we use the nonparametric Friedman's test, and its post hoc Nemenyi's test, we see that we get the same results as the sign test, with the only difference that ADA and ACC have no significant difference.

In Table 17, we can see the average and standard deviation of number of classifiers used by the compared ensemble methods. We can see that, though ACC and ADA have comparable accuracies, ACC uses fewer classifiers. This is because ADA uses the same base classifier, and to create diversity, needs more classifiers, whereas ACC uses different base classifiers and benefits from the diversity of those.

### 5.5. Recipe for constructing an ensemble

In this section, we will discuss how to best choose algorithms and construct ensembles for best performance.[2] Different learning algorithms have different inductive biases, which make them better for different data sets. There are parametric or nonparametric methods, those which work in the original space and those which map to a new space and work there. Furthermore, some algorithms are advantageous in terms of memory, some in terms of classification speed. The linear perceptron, for example, is cheap and fast but assumes linear separability; $k$-nearest neighbor is more flexible but expensive. Support vector machines and multilayer perceptrons may be slow to train on large data sets. The best classifier hence depends on the constraints of the application, availability of data and how much memory and computation can be afforded.

[2] We would like to thank the anonymous reviewer for suggesting us to write this section.

**Table 16**
Pairwise comparison of accuracies (wins/losses over 38) of ensemble methods using $5 \times 2$ cv $F$-test.

| | Acc | Dt.lin | Ada | Bag | Kne |
|---|---|---|---|---|---|
| Acc | 0 | **10** | **11** | **14** | **19** |
| Dt.lin | 1 | 0 | 9 | **10** | **17** |
| Ada | 0 | 7 | 0 | **10** | **16** |
| Bag | 0 | 2 | 0 | 0 | 5 |
| Kne | 0 | 1 | 0 | 1 | 0 |

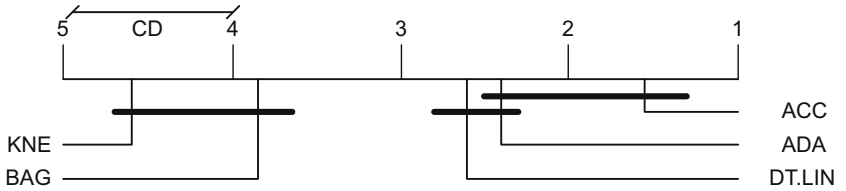


**Fig. 8.** Graphical representation of post hoc Nemenyi test results of compared methods with ranks given in Table 18.

**Table 17**
Average and standard deviation of number of base classifiers contained in different ensemble methods.

| Acc | Ada | Bag |
|---|---|---|
| $2.39 \mp 1.50$ | $22.10 \mp 8.11$ | $10.13 \mp 8.01$ |

In constructing ensembles, the most critical factor is to have classifiers that are accurate and diverse. Our experiments have shown that the best way for this is to use different algorithms. We have seen that this is more effective in generating a candidate set than using different feature subsets (for example, by the random space method), different samples of the data (for example, by bootstrapping), or different hyperparameters (for example, by combining multilayer perceptrons with different number of hidden units). Only if we are determined to use a single algorithm (for example, a decision tree), we can play with features/training sets/hyperparameters to generate a diverse and accurate set of versions of the single algorithm (a decision forest).

As the training set gets larger, different bootstraps will be more and more similar, there will be more positive correlation between classifiers and there will be less gain through bagging. Similarly, when there are few features, different feature subsets will generate more correlated classifiers. Furthermore, different algorithms may imply different constraints: For example, with $k$-nearest neighbor, different samples or different $k$ will not have a large effect on diversity, one should use different feature sets (or better still, $k$-nn should be combined with a completely different algorithm, for example, a linear perceptron).

In all this, we see the advantage of Icon, where we do not need to worry about how best to choose the classifiers in the ensemble; we train them all and let Icon pick the best subset.

We have also seen that it is better to combine classifiers with soft outputs (for example, posterior probabilities) rather than 0/1 decisions. In this work, we do not see a difference because we have well-trained classifiers with softmax outputs and the soft outputs are close to 0/1 anyway; but in general, soft outputs allow representing confidence and provide more information to the combiner.

If the base classifiers are all equally accurate and if there is no great difference among them in terms of accuracy, a fixed rule (for example, the sum rule) is cheap and accurate. In cases where there is difference, a trained rule works better as it allows assigning a small weight to bad classifiers. An ensemble construction method such as Icon also allows pruning inaccurate classifiers (which is equal to giving them zero weight). Note that a trained combiner needs its own training data which should be distinct from the data used to train the classifiers. If a model selection is to be done for the combiner, it also needs its own validation data; that is why, we have both *val-A* and *val-B* in this study.

If in a data set, we expect to have all classes with similar distributions then a classifier can be trusted for all classes. But this may not be true in a data set where certain classes are easily separated from others and some classes may be more difficult. In such a case, we would recommend using an ensemble of discriminants where a classifier is used for some classes but not all.

To summarize, classifier combination is not some magical tool which is always guaranteed to increase accuracy; it does increase memory and computational needs because there is the extra cost of training/storing/using many classifiers instead of one. Care should be taken that classifiers that are to be combined are both accurate and diverse and that they are combined appropriately; otherwise we will just pay without any significant gain.

**Table 18**
Average ranks of ensemble methods.

| Acc | Dt.lin | Ada | Bag | Kne |
|---|---|---|---|---|
| 1.55 | 2.61 | 2.42 | 3.84 | 4.58 |

## 6. Related work

Since choosing from a subset of classifiers is of exponential complexity, heuristic methods have been developed, which are similar to the ones we have evaluated in our study. Below, we divide work into categories, and give a chronological survey of literature for each category (Table 19 gives a summary). Since these categories have overlaps, we mention related work in the most appropriate subsection.

### 6.1. Subset selection algorithms

Besides the complexity and performance of evaluation of the ensemble, a subset of the given classifiers may outperform the whole set. So, methods that construct an ensemble using a subset of the available classifiers have been developed.

Partridge and Yates [41] use the concept of "methodological diversity" to come up with diverse classifiers and use three different methods for combining RBF or MLP neural networks. They use a subset selection technique, which first trains multiple base classifiers and chooses amongst them. The evaluations are carried out on three data sets, and no cross-validation is mentioned.

Ueda [56] combines multiple neural networks linearly to come up with an ensemble, similar to the work of Ting and Witten [55]. The approach is to first find best neural networks for each class and combine them linearly using optimal weights. Only neural network classifiers are used in the study, and weight decay regularization is used. The method is compared with voting, bagging and majority voting. Leave-one-out cross-validation is used for evaluating the results on one artificial and two real data sets; no statistical testing is provided.

Zhou et al. [65] show that choosing a subset of classifiers may be more accurate than combining all, and propose a genetic algorithm-based method to find the subset and evaluate their results on 10 regression and ten classification data sets. Their base classifiers are neural networks with one hidden layer. They compare their results with bagging and AdaBoost and show that their method finds ensembles which are smaller and which have better generalization ability. They use tenfold cross validation in their experiments and statistical tests are used to assess the performance.

Caruana et al. [10] propose an incremental algorithm that adds classifiers to the ensemble one at a time, as in Icon. The stopping conditions are different: They stop after a predefined number of steps or when all the classifiers are chosen. Their work uses thousands of models trained with different hyperparameters using six base algorithms, and they use seven data sets (which are binary or binarized) for evaluation purposes using ten performance metrics that are normalized to [0, 1].

Ruta and Gabrys [49] evaluate various methods for combining classifiers and compare error-based and diversity-based selection criteria. They make experiments on 27 data sets using 15 classifiers (different algorithms) using crisp (0/1) outputs. They evaluate various search methodologies including forward and backward search, single best and *m*-best, and evolution-

**Table 19**
Work similar to Icon analyzed in five dimensions: (1) Aim: SS: subset selection, EP: ensemble pruning, EC: ensemble construction, OWC: optimal weight. (2) Optimization criterion: A: accuracy, D: diversity, TP-FP: TP-FP spread, RCM: realistic cost, TC: test cost, AHR: average hit rate. (3) Base classifiers. (4) Optimization method: M: best *m*, G: genetic algorithm, F: forward, B: backward, R: random, L: floating, tabu search, X: exhaustive, E: early stop, O: optimization algorithms (integer programming, etc.), RLO: random linear oracle, C: clustering, DP: tree pruning, BC: best for class, MSE: squared error, MCE: classification error, MDM: margin minimization, CM: complementariness, CAL: calibration.

| Reference | Aim | Criteria | Base classifiers | Method | # of data sets |
|---|---|---|---|---|---|
| Margineantu and Dietterich [37] | EP | A, D | *dt* | L, F, E | 10 |
| Sharkey et al. [51] | SS | A | *ann*, *som* | R, X | 2 |
| Ueda [56] | SS, OWC | MCE, MSE | *ann* | BC | 3 |
| Roli et al. [48] | SS | A, D | *mlp*, *rbf*, *pnn*, *knn* | F, M, BC, B, L, C | 1 |
| Zhou et al. [65] | SS | A | *ann* | G | 20 |
| Caruana et al. [10] | SS | A, CAL | *svm*, *ann*, *knn*, *dt* | F | 7 |
| Ruta and Gabrys [49] | SS | A, D | *nmc*, *ann*, *knn*, *parzen*, *lp*, *rbf* | F, B, G | 27 |
| Demir and Alpaydın [12] | SS | A, TC | *mlp*, *lp* | F | 1 |
| Rokach et al. [47] | SS | AHR, QRecall, PEM | *dt* | M | 1 |
| Zhang et al. [64] | EP | A | *dt* | O | 16 |
| Kuncheva and Rodriguez [33] | EC | A | *dt* | RLO | 42 |
| Martínez-Munoz and Suárez [39] | SS | A | *dt* | M, E | 18 |
| Yang et al. [59] | SS, EC | A, TC | SPODE | F, B, R | 58 |
| This work | SS | A, D | *knn*, *mlp*, *svm*, *mdt*, *dt* | F, B, L | 38 |

ary algorithms. They use 16 measures (based on accuracy, diversity or both) to assess the performance of constructed systems, and their conclusion is the "inappropriateness of diversity measures used as selection criteria in favor of the direct combiner error-based search". They also conclude that "greedy algorithms are the best resistant to bad selection criteria". After the selection process they use majority voting for the combination of the classifiers. They divide the data set into 100 different train/test splits (half of test set is used for validation and the other half is used for evaluation). Their empirical results support our findings: It is best to use accuracy as the search criteria; using diversity alone as the selection criteria, which gives very bad results because it chooses most diverse but inaccurate classifiers. They use crisp (0/1) outputs, whereas using posterior probabilities as we do is more informative.

Kuncheva and Rodriguez [33] propose a hybrid selection–fusion approach to classifier combination. For each classifier, a random linear oracle is created as a hyperplane and the data residing in each half is trained using the ensemble approach. They test their method on 35 data sets from UCI and 7 other medical data sets. Tenfold cross validation was carried out with decision trees used as base classifiers. They see that all the ensemble methods benefit from this approach, with bagging and the random subspace methods having the highest benefits.

Sohn and Shin [52] compare bagging and combination using linear regression and see that on large data sets, they work equally well and bagging is better on small data sets (where probably a trained combiner overfits). They find that a trained combiner is suitable when there is strong correlation between input variables.

Yang et al. [59] combine Bayesian network classifiers and compare subset selection (in forward/backward direction using various criteria) vs. trained weighted combiner. They make their experiments on 58 benchmark data sets and use four different statistical testing methods for comparison. They conclude that there is no clear winner and that the choice between model selection and model weighing depends on the problem at hand.

### 6.2. Pruning algorithms

Some methods, instead of selecting a subset, begin with the whole set of classifiers and prune the unnecessary (according to an evaluation criterion) classifiers from the set to form the ensemble. Backward and tabu search in [48] and backward search in [49] are examples of this method.

Margineantu and Dietterich [37] use the idea that combining diverse classifiers leads to better ensembles: They first use boosting to form an ensemble of 50 decision trees from which they then prune (i.e., using backward search) non-diverse classifiers using five different diversity criteria. They use decision trees as base classifiers and they compare their results on ten data sets with tenfold cross-validation.

Sharkey et al. [51] propose the "test and select" approach to ensemble construction. The focus is on selecting a subset from a larger set. If the number of classifiers is small, they use exhaustive search; else they generate and test a number of candidates. One method is to generate random subsets, especially when there are too many base classifiers. They compare their results on two data sets and use neural networks for the first data set and self-organizing maps for the second data set. The data sets are divided into training, validation and testing sets, but statistical comparison is not mentioned.

Roli et al. [48] define the "overproduce and choose" paradigm where they initially construct a set of candidate classifiers and then select a subset amongst them. Their forward and backward search is the same as ours, using accuracy and diversity criteria; they also implement a forward version which does not start with the best classifier but a random one. They use only one data set and compare their results with ALL and BEST, with three different sets of base classifiers. No statistical evaluation is presented in the study and they combine using majority voting.

Tamon and Xiang [53] use a method called "weight shifting" to prune the ensembles of decision trees constructed using AdaBoost. They evaluate this method by using tenfold cross-validation on eight data sets, but no statistical testing is mentioned. They also show that subset selection problem is intractable, and propose a solution using integer programming.

Prodromidis and Stolfo [43] form a meta decision tree for the classifiers in an ensemble and use cost-complexity-based pruning to prune the tree. They first model the ensemble (constructed by any method) into a decision tree, which mimics the behavior of the original ensemble. They then prune this constructed tree, which results in the deletion of some of the classifiers in the original ensemble. The remaining classifiers are used to construct the final ensemble with the same method that is used to construct the original ensemble. The methods used for constructing the initial ensemble are weighted voting, majority voting, SCANN [40], and stacking, and five base classifiers were used in the study.

Bakker and Heskes [5] use clustering to reduce the size of bootstrap ensembles. They use cluster centers as representatives of each cluster and use a measure of methodological diversity to make the cluster centers optimally diverse. Islam et al. [23] use incremental algorithms based on the negative correlation learning to come up with neural network ensembles, taking into account both accuracy and diversity. They evaluate their proposed method on eight data sets. The diversity measure is added to the error function, so that networks try to maximize diversity during training.

Martínez-Munoz and Suárez [39] train $L$ classifiers using bagging, and then use AdaBoost to prune the ensemble (i.e. change the random order of bagging and early stop). They use 18 data sets in their experiments and use decision trees as base classifiers; and they also check for statistical improvement. Their conclusion is that, by selecting a subset of base classifiers, one can achieve better accuracy, with less complex ensembles. Their method outperforms bagging and is comparable to AdaBoost, though when the noise level is high, their method outperforms AdaBoost, which makes it a better alternative ensemble method when the noise level is not known. In their recent work, Martínez-Munoz et al. [38] compare methods for

selecting a subset of classifiers constructed with bagging. They compare different methods for changing the order of aggregation in bagging.

In a recent work, Dos Santos et al. [50] propose a dynamic overproduce and choose strategy which outperforms and is more efficient than a static "overproduce and select" strategy and a dynamic classifier selection method. They use random subspaces and bagging for overproduction phase, four different methods as search criteria, and use single and multiobjective genetic algorithms for optimization. The selection phase is divided into two phases: optimization and dynamic selection. For each test instance, a subset of classifiers is selected according to some confidence measure and they are combined.

### 6.3. Diversity-based algorithms

Kuncheva et al. [34] use nine different diversity measures and eight different combination techniques for bagging and boosting. The results indicate that there is no clear distinction between the diversity measures, and no clear relation between diversity and accuracy, but there are some interesting patterns.

Kuncheva and Whitaker [35] define ten different diversity measures and their relationship with classifier ensemble accuracy. They define ten different diversity measures used in the literature, some of which are closely related with one another, and classify them into groups such as pairwise and non-pairwise. At the end they conclude that there is no clear relationship between the ensemble accuracy and the diversity measures used.

Roli et al. [48] use diversity measures for evaluating the search algorithms used in subset selection. Goebel and Yan [18] use $\rho$-correlation-based diversity measure with an incremental learning algorithm to find an ensemble. They express the fact that $\rho$-correlation is a better indicator of contribution to overall accuracy than that of the individual accuracy of the base classifier that is to be added. They use neural network classifiers as base classifiers and they evaluate their results on one data set. No statistical testing is done.

A recent special journal issue edited by Kuncheva [32] contains papers on diversity measures and their role in ensemble construction: Banfield et al. [6] propose another diversity measure called the percentage correct diversity measure and use it for thinning (pruning).

Though most studies show that the relation between accuracy and diversity is unclear, Ko et al. [28,29] propose compound diversity measures, which combine accuracy and diversity, and show that these measures have high correlation with the ensemble accuracy.

### 6.4. Genetic and optimization algorithms

Kim et al. [26] train multiple ensembles using genetic algorithms. They evaluate their algorithm on 17 data sets using cross-validation and using neural networks as base classifiers. The diversity of the ensembles is achieved using multiple feature subsets. The difference of their method from most of the methods in the literature is that they build multiple ensembles at the same time. The genetic algorithm determines the membership of classifiers for the ensembles and the selected feature subsets of each classifier.

Zhang and Bhattacharyya [63] propose genetic programming (GP) to create ensembles particularly when there are much data at hand. A GP classifier consists of nodes that are of any possible mathematical operators and leaves that are of any features of the data. They create and compare mean combiners of genetic programming, decision tree and logistic regression classifiers, and report that the GP classifiers are more accurate than the other two. The possible reasons for this is the diversity created due to recombination processes (crossover and mutation) and the fitness function. They also note that since their GP classifiers evolve to contain only the features containing much information, there is also feature selection.

The fuzzy linguistic quantifiers can be used to represent the opinions of the individual decision makers [42]. There are two proposed methods to fuse the decisions. In one method, the ordered weighted averaging is used for obtaining a scalar value where the quantifiers help to determine the weights. In the other method, the majority opinion is represented as a vague concept, a fuzzy subset.

Zhang et al. [64] define the problem of pruning an ensemble as a quadratic integer programming problem and use semidefinite programming to get a better approximate solution.

### 6.5. Combining different representations

Perhaps the best way to form independent base classifiers is to have them use different representations of the same object or event. Different representations make different characteristics apparent and an object ambiguous in one representation may be clearly recognizable in another [1]. Demir and Alpaydın [12] generate multiple feature sets from different representations, feed them to separate base classifiers and incrementally find the best subset of classifier/representation pairs; different representations may have different costs associated with them and the chosen "cost conscious" ensemble best trades off accuracy and cost. They perform their experiments on a handwritten digit data set with multiple representations using multilayer perceptrons as base classifiers, and compare the accuracy with voting over all and boosting. Gökberk et al. [19] use different face representations.

## 7. Conclusions

We discuss two ensemble construction methods. In an ensemble of classifiers, we choose a subset from a larger set of base classifiers. In an ensemble of discriminants, we choose a subset of base discriminants, where a discriminant output of a base classifier by itself is assessed for inclusion in the ensemble.

A greedy, forward, incremental classifier ensemble finds ensembles that are small, as accurate as the optimal ensemble, and does this in polynomial time. It is best to maximize the final overall ensemble accuracy, rather than some intermediate diversity criterion; one may also envisage some combination of accuracy and diversity to be able to get the best of both worlds. We see that the incremental ensemble has statistically significantly higher accuracy than bagging and random subspace method; and when compared to AdaBoost no statistical difference can be found. On the other hand, Acc uses fewer classifiers other than bagging and AdaBoost.

When the base classifiers are trained with enough data and are accurate, there is no need for a trained linear combiner, and the fixed sum rule works as well, in a cheaper manner. Our experience in this and other studies is that stacking works better than voting when there is disparity between the accuracies of the base classifiers. When all base classifiers are accurate and equally trustworthy, voting works fine; stacking is needed when we need to weight some, those that are more accurate, higher and some, the erroneous ones, less.

The discriminant ensemble is interesting in the sense that not all discriminants of chosen classifiers are used. In many real-world data mining applications, training is typically done much less frequently than testing and decreasing test complexity at the expense of more training is preferred. The idea is the same as a classifier ensemble: just as not all base classifiers may not be needed and it is better to weed out those not needed to keep complexity in check, it may not be necessary to use a base classifier for all classes. Considering a classifier not as a single entity but as a collection of discriminants, one for each class, we can choose a subset of the discriminants thereby using a base classifier for some classes but not all. The corresponding discriminant is included, when the inductive bias of that classifier matches for a class, otherwise it is not used. This makes the whole ensemble much simpler, faster, and interpretable. In large scale data mining applications, where both accuracy and time/space complexity are important, such improvements in efficiency may be critical.

## Acknowledgements

We would like to thank the three anonymous referees and the editor for their constructive comments, pointers to related literature, and pertinent questions which allowed us to better situate our work as well as organize the ms and improve the presentation. This work has been supported by the Turkish Academy of Sciences in the framework of the Young Scientist Award Program (EA-TÜBA-GEBİP/2001-1-1), Boğaziçi University Scientific Research Project 05HA101 and Turkish Scientific Technical Research Council TÜBİTAK EEEAG 104E079.

## References

[1] F. Alimoğlu, E. Alpaydın, Combining multiple representations and classifiers for pen-based handwritten digit recognition, in: Proceedings of the International Conference on Document Analysis and Recognition, ICDAR'97, 1997, pp. 637–641.
[2] F.M. Alkoot, J. Kittler, Experimental evaluation of expert fusion strategies, Pattern Recognition Letters 20 (11–13) (1999) 1361–1369.
[3] E. Alpaydın, Combined $5 \times 2$ cv $F$ test for comparing supervised classification learning algorithms, Neural Computation 11 (8) (1999) 1885–1892.
[4] A. Asuncion, D.J. Newman, UCI machine learning repository, 2007. <http://www.ics.uci.edu/~mlearn/MLRepository.html>.
[5] B. Bakker, T. Heskes, Clustering ensembles of neural network models, Neural Networks 16 (2) (2003) 261–269.
[6] R.E. Banfield, L.O. Hall, K.W. Bowyer, W.P. Kegelmeyer, Ensemble diversity measures and their application to thinning, Information Fusion 6 (1) (2005) 49–62.
[7] A. Bertoni, R. Folgieri, G. Valentini, Bio-molecular cancer prediction with random subspace ensembles of support vector machines, Neurocomputing 63 (2005) 535–539.
[8] L. Breiman, Bagging predictors, Machine Learning 24 (2) (1996) 123–140.
[9] J.B.D. Cabrera, On the impact of fusion strategies on classification errors for large ensembles of classifiers, Pattern Recognition 39 (2006) 1963–1978.
[10] R. Caruana, A. Niculescu-Mizil, G. Crew, A. Ksikes, Ensemble selection from libraries of models, in: Proceedings of the International Conference on Machine Learning, ICML'04, 2004, pp. 137–144.
[11] C.-C. Chang, C.-J. Lin, LIBSVM: a library for support vector machines, 2001. <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
[12] C. Demir, E. Alpaydın, Cost-conscious classifier ensembles, Pattern Recognition Letters 26 (14) (2005) 2206–2214.
[13] J. Demsar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1–30.
[14] T.G. Dietterich, Approximate statistical tests for comparing supervised classification learning algorithms, Neural Computation 10 (7) (1998) 1895–1923.
[15] R.P. Duin, The combining classifier: to train or not to train?, in: Proceedings of the International Conference on Pattern Recognition, ICPR'02, 2002, pp. 765–770.
[16] Y. Freund, R.E. Schapire, Experiments with a new boosting algorithm, in: Proceedings of the International Conference on Machine Learning, ICML'96, 1996, pp. 148–156.
[17] G. Fumera, F. Roli, A. Serrau, A theoretical analysis of bagging as a linear combination of classifiers, IEEE Transactions on Pattern Analysis Machine Intelligence 30 (7) (2008) 1293–1299.
[18] K.F. Goebel, W. Yan, Choosing classifiers for decision fusion, in: P. Svensson, J. Schubert (Eds.), Proceedings of the International Conference on Information Fusion, Fusion'04, vol. I, 2004, pp. 563–568.
[19] B. Gökberk, H. Dutağacı, A. Ulaş, L. Akarun, B. Sankur, Representation plurality and fusion for 3D face recognition, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 38 (1) (2008) 155–173.
[20] J.V. Hansen, Combining predictors: comparison of five meta machine learning methods, Information Sciences 119 (1–2) (1999) 91–105.
[21] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning, Springer-Verlag, 2001.

[22] T.K. Ho, The random subspace method for constructing decision forests, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (8) (1998) 832–844.
[23] M. Islam, X. Yao, K. Murase, A constructive algorithm for training cooperative neural network ensembles, IEEE Transactions on Neural Networks 14 (2003) 820–834.
[24] R.A. Jacobs, M.I. Jordan, S.J. Nowlan, G.E. Hinton, Adaptive mixtures of local experts, Neural Computation 3 (1991) 79–87.
[25] A. Jain, K. Nandakumar, A. Ross, Score normalization in multimodal biometric systems, Pattern Recognition 38 (2005) 2270–2285.
[26] Y. Kim, W.N. Street, F. Menczer, Meta-evolutionary ensembles, in: Proceedings of the International Joint Conference on Neural Networks, IJCNN'02, vol. 3, 2002, pp. 2791–2796.
[27] J. Kittler, M. Hatef, R.P. Duin, J. Matas, On combining classifiers, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (3) (1998) 226–239.
[28] A.H.-R. Ko, R. Sabourin, A. de Souza Britto Jr., Combining diversity and classification accuracy for ensemble selection in random subspaces, in: Proceedings of the International Joint Conference on Neural Networks (IJCNN'06), 2006, pp. 2144–2151.
[29] A.H.-R. Ko, R. Sabourin, A. de Souza Britto Jr., A new objective function for ensemble selection in random subspaces, in: Proceedings of the International Conference on Pattern Recognition (ICPR'06), 2006, pp. 185–188.
[30] L.I. Kuncheva, A theoretical study on six classifier fusion strategies, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (2) (2002) 281–286.
[31] L.I. Kuncheva, Combining Pattern Classifiers: Methods and Algorithms, Wiley–Interscience, 2004.
[32] L.I. Kuncheva, Special issue on diversity in multiple classifier systems, Information Fusion 6 (1) (2005) 1–115.
[33] L.I. Kuncheva, J.J. Rodriguez, Classifier ensembles with a random linear oracle, IEEE Transactions on Knowledge and Data Engineering 19 (4) (2007) 500–508.
[34] L.I. Kuncheva, M. Skurichina, R.P.W. Duin, An experimental study on diversity for bagging and boosting with linear classifiers, Information Fusion 3 (4) (2002) 245–258.
[35] L.I. Kuncheva, C.J. Whitaker, Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy, Machine Learning 51 (2) (2003) 181–207.
[36] C.-L. Liu, Classifier combination based on confidence transformation, Pattern Recognition 38 (2005) 11–28.
[37] D.D. Margineantu, T.G. Dietterich, Pruning adaptive boosting, in: Proceedings of the International Conference on Machine Learning, ICML'97, 1997, pp. 211–218.
[38] G. Martínez-Munoz, D. Hernández-Lobato, A. Suárez, An analysis of ensemble pruning techniques based on ordered aggregation, IEEE Transactions on Pattern Analysis Machine Intelligence 31(2) (2009) 245–259.
[39] G. Martínez-Munoz, A. Suárez, Using boosting to prune bagging ensembles, Pattern Recognition Letters 28 (1) (2007) 156–165.
[40] C.J. Merz, Using correspondence analysis to combine classifiers, Machine Learning 36 (1–2) (1999) 33–58.
[41] D. Partridge, W.B. Yates, Engineering multiversion neural-net systems, Neural Computation 8 (4) (1996) 869–893.
[42] G. Pasi, R.R. Yager, Modeling the concept of majority opinion in group decision making, Information Sciences 176 (4) (2006) 390–414.
[43] A.L. Prodromidis, S.J. Stolfo, Cost complexity-based pruning of ensemble classifiers, Knowledge and Information Systems 3 (4) (2001) 449–469.
[44] C.E. Rasmussen, R.M. Neal, G. Hinton, D. van Camp, M. Revow, Z. Ghahramani, R. Kustra, R. Tibshirani, Delve data for evaluating learning in valid experiments, 1995–1996. <http://www.cs.toronto.edu/~delve/>.
[45] S. Raudys, Trainable fusion rules. I: Large sample size case, Neural Networks 19 (2006) 1506–1516.
[46] S. Raudys, Trainable fusion rules. II: Small sample-size effects, Neural Networks 19 (2006) 1517–1527.
[47] L. Rokach, O. Maimon, R. Arbel, Selective voting: getting more for less in sensor fusion, International Journal of Pattern Recognition and Artificial Intelligence 20 (3) (2006) 329–350.
[48] F. Roli, G. Giacinto, G. Vernazza, Methods for designing multiple classifier systems, in: Proceedings of the International Workshop on Multiple Classifier Systems, MCS'01, 2001, pp. 78–87.
[49] D. Ruta, B. Gabrys, Classifier selection for majority voting, Information Fusion 6 (1) (2005) 63–81.
[50] E.M.D. Santos, R. Sabourin, P. Maupin, A dynamic overproduce-and-choose strategy for the selection of classifier ensembles, Pattern Recognition 41 (2008) 2993–3009.
[51] A.J.C. Sharkey, N.E. Sharkey, U. Gerecke, G.O. Chandroth, The "test and select" approach to ensemble combination, in: Proceedings of the International Workshop on Multiple Classifier Systems, MCS'00, vol. 1857, 2000, pp. 30–44.
[52] S.Y. Sohn, H.W. Shin, Experimental study for the comparison of classifier combination methods, Pattern Recognition 40 (2007) 33–40.
[53] C. Tamon, J. Xiang, On the boosting pruning problem, in: Proceedings of the European Conference on Machine Learning, ECML'00, 2000, pp. 404–412.
[54] D.M.J. Tax, M. van Breukelen, R.P. Duin, J. Kittler, Combining multiple classifiers by averaging or multiplying?, Pattern Recognition 33 (9) (2000) 1475–1485
[55] K.M. Ting, I.H. Witten, Issues in stacked generalization, Journal of Artificial Intelligence Research 10 (1999) 271–289.
[56] N. Ueda, Optimal linear combination of neural networks for improving classification performance, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (2) (2000) 207–215.
[57] A. Ulaş, Incremental construction of cost-conscious ensembles using multiple learners and representations in machine learning, Ph.D. Thesis, Boğaziçi University, 2008.
[58] D.H. Wolpert, Stacked generalization, Neural Networks 5 (1992) 241–259.
[59] Y. Yang, G.I. Webb, J. Cerquides, K.B. Korb, J. Boughton, K.M. Ting, To select or to weigh: a comparative study of linear combination schemes for superparent-one-dependence estimators, IEEE Transactions on Knowledge and Data Engineering 19 (12) (2007) 1652–1665.
[60] O.T. Yıldız, E. Alpaydın, Linear discriminant trees, in: Proceedings of the International Conference on Machine Learning, ICML'00, 2000, pp. 1175–1182.
[61] O.T. Yıldız, A. Ulaş, M. Semerci, E. Alpaydın, AYSU: machine learning data sets for model combination, 2007. <http://www.cmpe.boun.edu.tr/~ulas/aysu>.
[62] C.-X. Zhang, J.-S. Zhang, RotBoost: a technique for combining rotation forest and AdaBoost, Pattern Recognition Letters 29 (10) (2008) 1524–1536.
[63] Y. Zhang, S. Bhattacharyya, Genetic programming in classifying large-scale data: an ensemble method, Information Sciences 163 (1–3) (2004) 85–101.
[64] Y. Zhang, S. Burer, W.N. Street, Ensemble pruning via semi-definite programming, Journal of Machine Learning Research 7 (2006) 1315–1338.
[65] Z.-H. Zhou, J. Wu, W. Tang, Ensembling neural networks: many could be better than all, Artificial Intelligence 137 (2002) 239–263.