Pruning an Ensemble of Classifiers via Reinforcement Learning

Ioannis Partalas [*], Grigorios Tsoumakas and Ioannis Vlahavas

Department of Informatics, Aristotle University of Thessaloniki 54124 Thessaloniki, Greece

Abstract

This paper studies the problem of pruning an ensemble of classifiers from a Reinforcement Learning perspective. It contributes a new pruning approach that uses the Q-learning algorithm in order to approximate an optimal policy of choosing whether to include or exclude each classifier from the ensemble. Extensive experimental comparisons of the proposed approach against state-of-the-art pruning and combination methods show very promising results. Additionally, we present an extension that allows the improvement of the solutions returned by the proposed approach over time, which is very useful in certain performance-critical domains.

1 Introduction

A very active research area during the recent years involves methodologies and systems for the production and combination of multiple predictive models. Within the Machine Learning community this area is commonly referred to as *Ensemble Methods* [6].

Ensemble methods have traditionally been used for increasing the accuracy of single classification and regression models. Ensembles achieve higher predictive performance than individual models, mainly through the correction of their uncorrelated errors. Today, ensemble methods continue to play an important role in predictive analysis as they provide an appealing solution to several other problems:

[*] Corresponding author.
Email addresses: `partalas@csd.auth.gr` (Ioannis Partalas), `greg@csd.auth.gr` (Grigorios Tsoumakas), `vlahavas@csd.auth.gr` (Ioannis Vlahavas).

- *Scale inductive algorithms to large databases* [25]. Most inductive algorithms are too computationally complex and suffer from memory problems when applied to very large databases. A solution to this problem is to horizontally partition the database into smaller parts, train a predictive model in each of the smaller manageable part and combine the predictive models.
- *Learn from multiple physically distributed data sets* [23,32]. Often such data can't be collected to a single site due to privacy or size reasons. This problem can be overcome through the combination of multiple predictive models, each trained on a different distributed data set.
- *Learn from concept-drifting data streams* [27,33]. The main idea here is to maintain an ensemble of classifiers that are trained from different batches of the data stream. Combining these classifiers with a proper methodology can solve the problem of data expiration that occurs whenever the learning concept drifts.

Ensemble methods comprise two main phases. The first one concerns the production of the different models. An ensemble can be composed of either homogeneous or heterogeneous models. Models that derive from different executions of the same learning algorithm are called Homogeneous. Such models can be produced by injecting randomness into the learning algorithm or through the manipulation of the training instances, the input attributes and the model outputs [7]. Models that derive from running different learning algorithms on the same data set are called Heterogeneous. The second phase of an ensemble method concerns the combination of the models. Common methods here include Voting, Weighted Voting and Stacking [36].

Recent work [2,4,24,12,10,31,16–20,22,37], has considered an additional intermediate phase, called *ensemble pruning*, that deals with the reduction of the ensemble size prior to combination. Ensemble pruning is important for two reasons: *efficiency* and *predictive performance*. Having a very large number of models in an ensemble adds a lot of computational overhead. For example, decision tree models may have large memory requirements [17] and lazy learning methods have a considerable computational cost during execution. The minimization of run-time overhead is crucial in certain applications, such as in stream mining. Equally important is the second reason, predictive performance. An ensemble may consist of both high and low predictive performance models. The latter may negatively affect the overall performance of the ensemble. Pruning these models while maintaining a high diversity among the remaining members of the ensemble is typically considered a proper recipe for an effective ensemble.

The problem of pruning an ensemble of classifiers has been proved to be an NP-complete problem [29]. Exhaustive search for the best subset of classifiers is not tractable for ensembles that contain a large number of models. Greedy approaches, such as [2,4,17,18,20], are fast, as they consider a very small part of the space of all combinations. This however, may lead to suboptimal solutions of the pruning problem.

This work studies the problem of pruning an ensemble of classifiers from a Reinforcement Learning (RL) perspective. It uses the Q-learning algorithm in order to approximate an optimal policy of choosing whether to include or exclude each algorithm from the ensemble. The proposed algorithm visits a larger part of the state space than greedy algorithms, at the expense of larger execution time, but with the aim to discover a better solution.

The paper extends our previous work [22], with a new RL representation that is more general and models more efficiently the ensemble pruning problem. It also presents more extensive experiments and uses appropriate statistical tests for the evaluation of the results. Finally, it introduces an extension that allows the improvement of the solutions returned by the proposed approach over time, which is very useful in certain performance-critical domains. Experimental comparisons of the proposed approach against state-of-the-art pruning and combination methods show very promising results.

The rest of this paper is structured as follows: Section 2 presents background information on RL and Ensemble Methods. Section 3 reviews related work on pruning ensembles of classifiers, as well as on using RL to model problems related to pruning. Section 4 introduces the proposed approach. Section 5 presents the setup of the comparative experiments and Section 6 discusses the results. Section 7 presents the extension of the proposed approach for improving its performance over time, and finally Section 8 concludes this work.

2 Background

In this section we present background material on Reinforcement Learning (RL) and Ensemble Methods.

2.1 Reinforcement Learning

RL addresses the problem of how an agent can learn a behavior through trial-and-error interactions with a dynamic environment [28]. In an RL task the agent, at each time step t, senses the environment's state, $s_t \in S$, where S is the finite set of possible states, and selects an action $a_t \in A(s_t)$ to execute, where $A(s_t)$ is the finite set of possible actions in state s_t. The agent receives a reward, $r_{t+1} \in \Re$, and moves to a new state s_{t+1}. The general goal of the agent is to maximize the expected return, where the return, R_t, is defined as some specific function of the reward sequence.

The most widely-used model of optimal behavior for an RL agent is the infinite-horizon discounted model [14], which takes the long-run reward of the agent into account, but future rewards are geometrically discounted according to discount factor γ, $0 \leq \gamma < 1$:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

Some applications have a natural notion of a final time step [28] (e.g. end of a game). In these applications, the agent-environment interaction breaks naturally into subsequences, called *episodes*. Each episode ends in a special state called the *terminal state*, followed by a reset to a standard starting state or to a sample from a standard distribution of starting states. To accommodate the use of the infinite horizon discounted model, a terminal state is modeled as an *absorbing state*. An absorbing state has a single action that deterministically leads back to itself with a reward of 0. When the agent is guaranteed to reach a terminal state, then γ can be set to 1.

A *policy* π specifies that in state s the probability of taking action a is $\pi(s, a)$. For any policy π, the *state-value function*, $V^\pi(s)$, denotes the expected discounted return, if the agent starts from s and follows policy π thereafter. The value $V^\pi(s)$ of s under π is defined as:

$$V^\pi(s) = E_\pi \left\{ R_t \mid s_t = s \right\} = E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s \right\},$$

where s_t and r_{t+1} denote the state at time t and the reward received after acting at time t, respectively.

Similarly, the *action-value function*, $Q^\pi(s, a)$, under a policy π can be defined as the expected discounted return for executing a in state s and thereafter following π:

$$Q^\pi(s, a) = E_\pi \left\{ R_t \mid s_t = a, a_t = a \right\} = E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = a, a_t = a \right\}.$$

The optimal policy, π^*, is the one that maximizes the value, $V^\pi(s)$, for all states s, or the action-value, $Q^\pi(s, a)$, for all state-action pairs.

In order to learn the optimal policy, the agent learns the *optimal value function*, V^*, or the *optimal action-value function*, Q^* which is defined as the expected return of taking action a in state s and thereafter following the optimal policy π^*:

$$Q^*(s,a) = E\left\{r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a') \middle| s_t = s, a_t = a\right\}$$

The optimal policy can now be defined as:

$$\pi^* = \arg\max_a Q^*(s,a)$$

A widely used algorithm for finding the optimal policy is the Q-learning algorithm [34] which approximates the Q function with the following form:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)).$$

2.2 Ensemble Methods

2.2.1 Producing the Models

An ensemble can be composed of either homogeneous or heterogeneous models. Homogeneous models derive from different executions of the same learning algorithm by using different values for the parameters of the learning algorithm, injecting randomness into the learning algorithm or through the manipulation of the training instances, the input attributes and the model outputs [7]. Methods for producing homogeneous models are bagging [3] and boosting [26].

Heterogeneous models derive from running different learning algorithms on the same dataset. Such models have different views about the data, as they make different assumptions about it. For example, a neural network is robust to noise in contrast to a k-nearest neighbor classifier.

2.2.2 Combining the Models

A lot of different ideas and methodologies have been proposed in the past for the combination classification models. The main motivation behind this research is the common observation that there is no single classifier that performs significantly better in every classification problem. The necessity for high classification performance in some critical domains (e.g. medical, financial, intrusion detection) have urged researchers to explore methods that combine different classification algorithms in order to overcome the limitations of individual learning paradigms.

Unweighted and Weighted Voting are two of the simplest methods for combining not only Heterogeneous but also Homogeneous models. In Voting, each model outputs a class value (or ranking, or probability distribution) and the class with the most votes (or the highest average ranking, or average probability) is the one proposed by the ensemble. In Weighted Voting, the classification models are not treated equally. Each model is associated with a coefficient (weight), usually proportional to its classification accuracy.

Let x be an instance and m_i, $i = 1..k$ a set of models that output a probability distribution $m_i(x, c_j)$ for each class c_j, $j = 1..n$. The output of the (weighted) voting method $y(x)$ for instance x is given by the following mathematical expression:

$$y(x) = \arg\max_{c_j} \sum_{i=1}^{k} w_i m_i(x, c_j),$$

where w_i is the weight of model i. In the simple case of voting (unweighted), the weights are all equal to one, that is, $w_i = 1, i = 1..k$.

Stacked Generalization [36], also known as Stacking, is a method that combines multiple classifiers by learning a meta-level (or level-1) model that predicts the correct class based on the decisions of the base-level (or level-0) classifiers. This model is induced on a set of meta-level training data that are typically produced by applying a procedure similar to k-fold cross-validation on the training data: Let D be the level-0 training data set. D is randomly split into k disjoint parts $D_1 \ldots D_k$ of equal size. For each fold $i = 1 \ldots k$ of the process, the base-level classifiers are trained on the set $D \setminus D_i$ and then applied to the test set D_i. The output of the classifiers for a test instance along with the true class of that instance form a meta-instance. A meta-classifier is then trained on the meta-instances and the base-level classifiers are trained on all training data D. When a new instance appears for classification, the output of all base-level classifiers is first calculated and then propagated to the meta-level classifier, which outputs the final result.

In [9] Dzeroski and Zenko proposed a new method of stacking, where they use multi-response model trees at the meta-level. Experimental comparisons showed that stacking with mutli-response model trees performs better than other stacking approaches.

3 Related Work

This section reviews related work on the ensemble pruning problem. Additionally, we present approaches that adopt the RL framework to solve related problems to ensemble pruning.

Margineantu and Dietterich [17] introduce heuristics to calculate the benefit of adding a classifier to an ensemble, using forward selection in a number of them. These heuristics are based on the diversity and the performance of the classifiers. The authors experiment with boosting ensembles and conclude that pruning can help an ensemble to increase its predictive performance.

Fan et al. [10] prune an ensemble of classifiers using forward selection of the classification models, like in [17]. As a heuristic, they use the benefit that is obtained by evaluating the combination of the selected classifiers with the method of voting. Their results show that pruning increases the predictive performance and speeds up the run time of an ensemble of C4.5 decision trees trained on disjoint parts of a large data set.

Zhou and Tang [38] perform stochastic search in the space of model subsets using a standard genetic algorithm. The models in this case are decision trees produced through bagging. The ensemble is represented as a bit string, using one bit for each decision tree. Decision trees are included or excluded from the ensemble depending on the value of the corresponding bit. Standard genetic operations such as mutations and crossovers are used and default values are used for the parameters of the genetic algorithm. The voted performance of the ensemble is used as a function for evaluating the fitness of individuals in the population.

Caruana et al. [4] produce an ensemble of 1000 classification models using different algorithms and different sets of parameters for these algorithms. They subsequently prune the ensemble following an approach that is similar to [17]. This way they manage to achieve very good predictive performance compared to state-of-the-art ensemble methods.

Martinez-Munoz et al. [19,20] present two algorithms for pruning an ensemble of classifiers. In [19] the authors define for each classifier a vector with dimensionality equal to the size of the training set, where each element i corresponds to the decision of the classifier for the instance i. The classifier is added to the ensemble according to its impact in the difference between the vector of the ensemble (average of individual vectors) with a predefined reference vector. This reference vector indicates the desired direction towards which the vector of the ensemble must align. In [20], the authors produce an initial ensemble of bagging models. Then using a procedure based on boosting, they add to the ensemble the classifier with the lowest error and adapt the weights for the next classifier using the weights of the currently incorporated classifier. The process ends when a predefined size for the final pruned ensemble is reached.

Giacinto and Roli [12] employ Hierarchical Agglomerative Clustering (HAC) for classifier pruning. This type of clustering requires the definition of a distance metric between two data points (here classifiers). The authors defined this metric as the probability that the classifiers do not make coincident errors and estimate it from a validation set in order to avoid overfitting problems. The authors also defined the distance between two clusters as the maximum distance between two classifiers belonging to these clusters. This way they implicitly used the *complete link* method for inter-cluster distance computation. Pruning is accomplished by selecting a single representative classifier from each cluster. The representative classifier is the one exhibiting the maximum average distance from all other clusters.

Tsoumakas et al. [31,30] prune an ensemble of heterogeneous classifiers using statistical procedures that determine whether the differences in predictive performance among the classifiers of the ensemble are significant. Only the classifiers with significantly better performance than the rest are retained and subsequently combined with the methods of (weighted) voting. The obtained results are better than those of state-of-the-art ensemble methods.

Zhang et al. [37] formulate the ensemble pruning problem as a mathematical problem and apply semi-definite programming (SDP) techniques. In specific, the authors initially formulated the ensemble pruning problem as a quadratic integer programming problem that looks for a fixed-size subset of k classifiers with minimum misclassification and maximum divergence. They subsequently found that this quadratic integer programming problem is similar to the "max cut with size k" problem, which can be approximately solved using an algorithm based on SDP. Their algorithm requires the number of classifiers to retain as a parameter and runs in polynomial time.

3.2 RL approaches to related problems

Reinforcement Learning (RL) has not been used in the past for ensemble pruning. However, we found two approaches that use RL for solving two different yet related problems: classifier selection and algorithm (e.g. sorting) selection. We believe that it is worth mentioning them, because they offer some interesting alternative ideas on how can one use RL in order to model related problems that are not directly suitable for RL.

Dimitrakakis and Bengio [8] use RL to adapt a policy for the combination of multiple classifiers. Specifically, an architecture with n experts (classifiers), implemented by multi-layer perceptrons (MLPs) and an additional MLP with n outputs acting as the controlling agent are employed. The state space of the controlling agent consists of the instance space (all the possible different instances) of the particular classification problem and the action is the choice of the expert who will take the classification decision. On top of that, the expert who has been chosen uses the instance to train itself.

Lagoudakis and Littman [15], formulate the problem of algorithm selection as a Markov Decision Process (MDP) and use an RL approach to solve it. Given a set of algorithms that are equivalent in terms of the problem they solve, and a set of instance features, such as problem size, an RL approach is used to select the right algorithm for each instance based on the set of features. The state of the MDP is represented by the features of the current instance and the actions are the different algorithms that can be selected. Finally, the immediate cost for choosing some algorithm on some problem is the real time taken for that execution. The learning mechanism is a variation of the Q-learning algorithm.

4 Our Approach

We first formulate the problem of pruning an ensemble of classifiers $C = \{c_1, c_2, \ldots, c_n\}$ as a Reinforcement Learning (RL) task. We then present the particular RL algorithm used to deal with the problem.

4.1 Ensemble pruning as an RL task

We define the following components of every RL task, in terms of the ensemble pruning problem:

(1) A set of states, S.
(2) A set of actions, A.
(3) A reward function, $r(s, a)$.

In our approach a state s is a pair (C', c_i). C' is the current ensemble, a subset of C containing the classifiers that have been so far selected for inclusion in the final pruned ensemble, while c_i is the classifier that is currently under evaluation and will be included or excluded based on the next action. Consequently, the set of states S is the Cartesian product of C and its powerset, $P(C)$:

$$S = P(C) \times C$$

In each state (C', c_i) the agent can select between two actions. It can either include algorithm c_i into the current ensemble or not. Therefore A contains the $2n$ actions of including and excluding each algorithm:

$$A = \bigcup_{i=1}^{n} \{include(c_i), exclude(c_i)\}$$

The task of selecting a subensemble of classifiers is modeled as an episodic task, where each episode proceeds as follows: it starts with an empty set of classifiers, $s_0 = (\emptyset, c_1)$, and lasts n time steps. At each time step, $t = 1 \ldots n$, the agent chooses to include or not classifier c_t into the ensemble, $A(s_{t-1}) = \{include(c_t), exclude(c_t)\}$. The episode ends when the decision to include or exclude the n^{th} algorithm is taken and the agent arrives at the final state s_n. The presentation order of the classifiers during the episode is fixed and it does not change from one iteration to the next. Figure 1 graphically shows an episode. Since the agent is guaranteed to visit the terminal state, we set γ to 1.

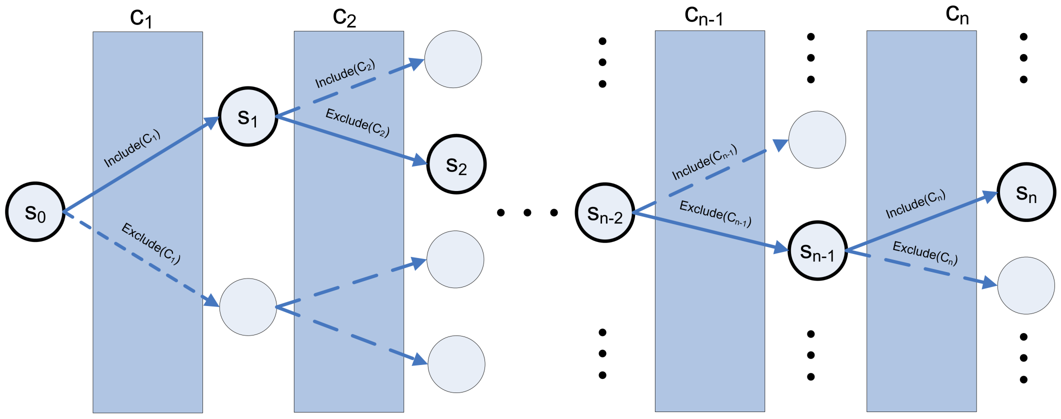


Fig. 1. The procedure of selecting a sequence of classifiers.

Rewards are equal to zero for all transitions, apart from the final one, where the agent receives a reward equal to the predictive performance of the ensemble of the final state. The reward definition is intentionally general in order to stress the capability of using different instantiations of the performance metric, the performance evaluation method and the classifier combination approach, depending on the requirements of the domain or the preferences of the data analyst.

For example, the performance evaluation metric could be instantiated to accuracy, area under the ROC curve, precision/recall f-score, a cost-sensitive metric and others. Similarly, cross-validation, repeated hold-out or some other method could be used for performance evaluation. For the combination of the classifiers, simple and fast approaches such as voting are suggested.

The above RL modeling of the pruning problem aims to maximize the performance of the final pruned ensemble, by rewarding the agent only on the final step. It restricts the actions to a small number (two for each state) and consequently reduces the overall complexity of the proposed algorithm. This offers the advantage of requiring less episodes to train the agent.

4.2 The proposed algorithm

We now continue with the particular RL algorithm that we propose in order to deal with the above problem, which is based on Q-Learning.

During an episode, the agent must stochastically select actions in order to explore the state space. One way to achieve this aim is to make use of the $\epsilon - greedy$ action selection method, where an action a is selected according to the following rule:

$$a = \begin{cases} \text{a random action with probability } \epsilon \\ \arg\max_{a'} Q(s, a') \text{ with probability } 1 - \epsilon \end{cases}$$

In this work we employ function approximation methods to tackle the problem of the large state space as well as the time needed to fill the values for every state-action pair in a tabular policy format. A well-known method is the combination of Q-learning with eligibility traces, $Q(\lambda)$, and gradient descent function approximation [28]. Eligibility traces can be considered as a measure of the visiting frequency of a state and are used to speed up the training process. Each state-action pair is associated with its eligibility trace, which is updated in order to give credit to actions.

Additionally, linear methods are used to approximate and represent the value function. In linear methods the value function $Q_t(s, a)$ is a linear function of a parameter vector $\vec{\theta}_t$ whose number of parameters is equal to the number of features in a state. At the training phase, a linear network receives an input vector which represents the features of the current state and its output is an estimation of the action-value of the state. Gradient descent methods are used to update the weights of the network. Further details about the algorithms can be found in [28].

The feature vector of the state has length $n + 1$, where the first n coordinates represent the presence or absence of the classifiers from the ensemble and the last coordinate represents the classifier that is being tested.

Figure 2 presents the pseudocode of the proposed approach. At the end of the training phase, the agent executes a final episode choosing the action with the highest Q value at each time step. The resulting subset of classifiers is the output of our approach.

```
(1) Initialize $\overrightarrow{\theta}$, $\overrightarrow{e_0} = \overrightarrow{0}$, $\epsilon$ and feature vector $\overrightarrow{\phi} = \overrightarrow{0}$
(2) Repeat (for each episode):
    (a) $s \leftarrow (\emptyset, c_1)$
    (b) Estimate $Q(s, include(c_1))$ and $Q(s, exclude(c_1))$
    (c) For each algorithm $c_i$
        (i) $p \leftarrow RandomReal(0, 1)$
        (ii) If $p < \epsilon$ then
            (A) $a \leftarrow$ random action
            (B) $\overrightarrow{e_0} = \overrightarrow{0}$
        (iii) Else
            (A) $a \leftarrow \arg\max_{a'} Q(s, a')$
            (B) $\overrightarrow{e} = \lambda \overrightarrow{e}$
    (d) $\overrightarrow{e} = \overrightarrow{e} + \phi_s$
    (e) Estimate $Q(s, a), \forall a \in A(s')$
        - $Q(s, a) = \sum_{i=1}^{n+1} \theta(i)\phi_s(i)$
    (f) $a' \leftarrow \arg\max_a Q_a$
    (g) $\delta \leftarrow r + \arg\max_{a'} Q_{a'} - Q_a$
    (h) $\overrightarrow{\theta} \leftarrow \overrightarrow{\theta} + \alpha\delta\overrightarrow{e}$
    (i) Set $s \leftarrow s'$
(3) Until $\overrightarrow{\theta}$ converges.
```

Fig. 2. Pseudocode of the proposed approach.

5 Experimental Setup

The empirical comparison of EPRL against other combination and pruning methods is based on 20 data sets from the UCI Machine Learning repository [1]. Table 1 presents the details of these data sets (Folder in UCI server, number of instances, classes, continuous and discrete attributes, percentage of missing values).

Initially each dataset is randomly split into three disjunctive parts: a training set D_{Tr}, an evaluation set D_{Ev} and a test set D_{Te}, consisting of 60%, 20% and 20% of the examples in the dataset respectively. Two different ensemble production methods are used to create an ensemble of 100 models based on the data of D_{Tr}: a) running different learning algorithms with different parameter configurations (heterogeneous ensemble), and b) bootstrap sampling, as in bagging (homogeneous ensemble).

The WEKA machine learning library is used as the source of the learning algorithms [35] in both cases. In the second case we train 100 decision trees using the C4.5 algorithm with default configuration (pruning with 0.25 confidence factor). In the first case we train 2 naive Bayes classifiers, 4 decision trees, 32 multilayer perceptrons, 32 k-Nearest Neighbors (k-NNs) and 30 support vector machines (SVMs). The different parameters that are used to train

Table 1
Details of the data sets: Folder in UCI server, number of instances, classes, continuous and discrete attributes, percentage of missing values

UCI Folder	Inst	Cls	Cnt	Dsc	MV(%)
audiology	226	24	0	69	2.03
breast-cancer	286	2	0	9	0.35
breast-cancer-winsonsin	699	2	9	0	0.25
chess (kr-vs-kp)	3196	2	0	36	0.00
cmc	1473	3	2	7	0.00
dermatology	366	6	1	33	0.01
ecoli	336	8	7	0	0.00
glass	214	7	9	0	0.00
heart-disease (hungary)	294	5	6	7	20.46
heart-disease (switzerland)	123	5	6	7	17.07
hepatitis	155	2	6	13	5.67
image	2310	7	19	0	0.00
ionosphere	351	2	34	0	0.00
iris	150	3	4	0	0.00
labor	57	2	8	8	35.75
lymphography	148	4	3	15	0.00
pima-indians-diabetes	768	2	9	0	0.00
statlog (australian)	690	2	6	9	0.65
statlog (german)	1000	2	7	13	0.00
statlog (heart)	270	2	13	0	0.00

the algorithms are the following (default values are used for the rest of the parameters):

- Naive Bayes: we built one model with default parameters and one with kernel estimation.
- Decision trees: we used 2 values for the confidence factor {0.25, 0.5}, and 2 values for Laplace smoothing {true, false}.
- Multilayer perceptrons: we used 4 values for the hidden nodes {1, 2, 4, 8}, 2 values for the learning rate {0.3, 0.6} and 4 values for the momentum term {0.0, 0.2, 0.5, 0.9}.
- k-NNs: we used 16 values for k distributed evenly between 1 and the number

of training instances. We also used 2 weighting methods: {no-weighting, inverse weighting}.

- SVMs: we used 3 values for the complexity parameter $\{10^{-5}, 10^{-4}, 10^{-3}\}$ and 10 different kernels. We used 2 polynomial kernels (degree 2 and 3) and 8 radial kernels (gamma {0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 2}).

We compare the performance of our approach, Ensemble Pruning via Reinforcement Learning (EPRL), against the classifier combination methods *Voting* (V) and *Stacking with Multi-Response Model Trees* (SMT) [9] and the ensemble pruning methods *Forward Selection* (FS) [10,4] and *Selective Fusion* (SF) [30]. SF is a method for pruning ensembles of heterogeneous classifiers and does not participate in the experiments concerning homogeneous ensembles.

EPRL is executed until the difference in the weights of the linear network between two subsequent episodes becomes less than a small threshold, 10^{-4}. This eliminates the need to specify explicitly a number of episodes. The performance of the pruned ensemble at the end of each episode is evaluated on D_{Ev} based on its accuracy using voting. The models of the final subensemble are combined using voting. The value of ϵ was set to 0.6, in order to have a high degree of exploration, and reduced by a factor of 0.0001% at each episode. The weighting factor λ is set to 0.9 to speed up the training process. Note that the specific values of all these hyperparameters may be influencing the results of the experiments, so conclusions should be generalized with caution. However, we believe that changes to ϵ and λ parameters will mostly affect the training time and not the overall performance of the method.

In SMT, D_{Ev} is used for the production of the required meta-level training data. In the combination method V the set $D_{Tr} \cup D_{Ev}$ is used to build the ensembles.

FS starts with an empty ensemble and at each step it greedily adds the classifier that leads to the highest accuracy of the ensemble on D_E using the voting method. From the sequence of subensembles generated, we select the subensemble with the highest classification accuracy on S_{Ev} using voting, as in [4], instead of using an arbitrary percentage or number of models. This stopping procedure is the original that used in [4]. The models of the selected subensemble are combined using voting.

SF has three tunable parameters: the multiple comparisons procedure, the confidence interval of this procedure and the combination method for the selected subensemble. We use Tukey's test for the first parameter, as it has been found to outperform two other procedures (Hsu's test, Scott & Knott's procedure) in a past study [30]. We set the confidence interval to the standard value used in most statistical procedures (95%), leading to a critical value of 8.317. The critical value was calculated via regression, as we didn't find a table of precalculated critical values for 100 treatments. The set $D_{Tr} \cup D_{Ev}$ is used to train the base models of the ensembles. The models of the selected subensemble are combined using voting, similarly to EPRL and FS.

The performance of all methods is evaluated on D_{Te}. Apart from the classification accuracy, we also record the size of the final ensemble for the ensemble pruning methods. The whole experiment is performed 10 times for each dataset and the results are averaged.

6 Results and Discussion

In this section we discuss and analyze the results for both the heterogeneous and the homogeneous case.

6.1 Heterogeneous case

Table 2 presents the accuracy along with the standard deviation of each algorithm on each dataset, as well as the corresponding ranks. With bold typeface, we highlight the winning algorithm in each dataset. We first notice, that the proposed algorithm obtains the best performance in 7 cases followed by SF and SMT, with 5 winning cases each, while FS and V have the best performance in 4 and 2 cases respectively. This fact highlights the strength of the proposed approach.

According to Demsar [5], the appropriate way to compare multiple algorithms on multiple data sets is based on their average rank across all data sets. On each data set, the algorithm with the highest accuracy gets rank 1.0, the one with the second highest accuracy gets rank 2.0 and so on. In case two or more algorithms tie, then they all receive the average of the ranks that correspond to them.

We notice that the best performing algorithm is EPRL with average rank 2.475 while SF, FS, V and SMT follow up with average ranks 2.5, 2.625, 3.675 and 3.725 respectively. The fact that the proposed approach is on average the best performing algorithm as well as that it is ranked in the first place for 7 datasets out of 20, shows not only its strength but also its robustness. In contrast, the method with the second best rank (SF) achieves the highest accuracy (and rank) in only five datasets.

Next, we make use of Friedman's test [11] that compares the average ranks of the algorithms under the null-hypothesis, which states that all algorithms are equivalent and so must be their performance. More specifically, we use the F_F test that was proposed by [13] and is based on Friedman's χ_F^2 statistic. With confidence level $p < 0.05$ the F_F test shows critical differences among the algorithms and so we reject the null hypothesis and proceed to the post-hoc Nemenyi test [21].

Figure 3 graphically represents the results of the Nemenyi test with 90% confidence, $q_{0.10} = 2.459$ and $CD = 1.23$. The best ranks are to the right and the groups of algorithms that are not significantly different are connected with a bold line. We notice that there are three groups of similar algorithms and that EPRL is significantly better than SMT and V.

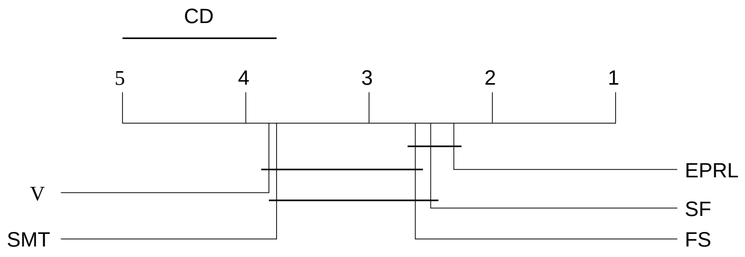


Fig. 3. Graphical representation of the Nemenyi test for the Heterogeneous case.

As far as concerns the performance of V, we observe in Table 2 that the use of all the members of the ensemble, leads to low performance. On the other hand, the pruning algorithms outperform the whole ensemble and in particular the accuracy is increased fair enough in most cases. This fact justifies the pruning procedure.

Another interesting observation is the bad performance of SMT. This behavior can be explained if we take into account the fact that SMT can't handle problems with many base-level classifiers due to its inefficient to generalize well from the meta-level training data for a large ensemble size.

Table 3 depicts the size of the pruned ensemble of each algorithm on each dataset along with the average values across all datasets. FS and EPRL produce the smallest final ensembles with an average of 5.0 and 5.47 models respectively, while SF keeps 73.55 models at average. Considering also the performance of the algorithms, we conclude that EPRL achieves the best performance and keeps the size of the final ensemble small.

Figures 4(a), 4(b) and 4(c) present the average type of models selected from FS, EPRL and SF respectively. The results are averaged for all datasets. We first notice that all the algorithms select mostly MLPs and $k - NNs$. More specifically, SF selects a balanced mixture of MLPs (28.2) and $k - NNs$ (24.9) and interestingly in all cases it keeps the DT models (4.0). EPRL and FS produce ensembles with 2.8 and 3.1 MLPs along with 1.5 and 1.2 $k - NNs$ respectively.

Table 2
Folder in UCI server, accuracy and rank of each method on each of the 20 datasets for the heterogeneous case.

	Accuracy					Rank				
UCI Folder	FS	EPRL	SF	V	SMT	FS	EPRL	SF	V	SMT
audiology	77.3±4.0	**78.0**±4.7	77.8±5.9	75.9±6.1	26.4±5.3	3.0	1.0	2.0	4.0	5.0
breast-cancer	**74.4**±4.8	73.3±4.6	71.6±4.2	71.6±4.2	66.5±4.7	1.0	2.0	3.5	3.5	5.0
breast-w	96.3±1.5	96.3±1.6	96.9±1.8	95.0±1.9	**97.5**±2.1	3.5	3.5	2.0	5.0	1.0
cmc	52.8±2.4	**53.2**±2.7	51.6±4.5	47.1±2.7	45.5±3.6	2.0	1.0	3.0	4.0	5.0
dermatology	96.6±1.5	**96.7**±1.5	96.5±1.0	96.4±1.3	65.3±2.2	2.0	1.0	3.0	4.0	5.0
ecoli	**83.9**±4.3	82.8±4.8	83.7±5.0	82.4±5.2	67.2±6.1	1.0	3.0	2.0	4.0	5.0
kr-vs-kp	99.3±0.3	99.2±0.2	**99.4**±0.2	98.8±0.5	97.6±0.5	2.0	3.0	1.0	4.0	5.0
glass	68.1±5.7	**70.2**±6.4	68.6±5.5	68.1±5.5	52.1±7.2	3.5	1.0	2.0	3.5	5.0
heart-h	79.5±5.4	79.0±5.7	79.9±5.6	79.9±5.6	**80.7**±6.3	4.0	5.0	2.5	2.5	1.0
hepatitis	81.3±5.9	81.3±5.9	78.1±4.0	78.1±4.0	**81.9**±5.9	2.5	2.5	4.5	4.5	1.0
image	96.6±0.6	96.8±0.6	**97.0**±0.5	96.2±0.8	64.0±1.0	3.0	2.0	1.0	4.0	5.0
ionosphere	**91.6**±3.0	**91.6**±3.0	90.7±3.3	83.4±3.2	85.3±3.1	1.5	1.5	3.0	5.0	4.0
iris	94.7±0.4	94.7±0.4	95.7±3.3	94.0±2.4	**99.3**±1.3	3.5	3.5	2.0	5.0	1.0
labor	89.1±8.9	89.1±8.9	**94.5**±4.5	**94.5**±4.5	83.6±7.8	3.5	3.5	1.5	1.5	5.0
lymph	82.4±4.4	80.3±4.3	**85.5**±4.8	**85.5**±4.8	78.3±6.1	3.0	4.0	1.5	1.5	5.0
diabetes	75.2±4.1	**75.7**±3.9	67.5±6.1	66.5±4.6	75.2±4.7	2.5	1.0	4.0	5.0	2.5
credit-a	85.1±1.5	85.5±2.4	**85.7**±2.2	83.8±2.3	83.6±3.5	3.0	2.0	1.0	4.0	5.0
credit-g	73.2±2.6	**74.4**±2.2	69.0±2.4	69.0±2.4	69.8±2.6	2.0	1.0	4.5	4.5	3.0
heart-statlog	**82.2**±5.6	81.9±6.2	81.5±4.3	81.5±3.5	79.1±4.2	1.0	2.0	3.5	3.5	5.0
heart-s	33.3±9.3	32.9±8.6	37.5±8.5	37.5±8.5	**41.3**±8.4	4.0	5.0	2.5	2.5	1.0
Average	80.64	80.64	80.43	79.31	72.01	2.575	2.425	2.5	3.775	3.725

6.2 Homogeneous case

Table 4 shows the accuracy, along with the standard deviation, and the corresponding rank of each algorithm on each dataset for the homogeneous models. We notice that the best performing algorithm is V, 1.225, and follows EPRL and FS with 2.05 and 2.825 average rank respectively. EPRL is the best performing algorithm in 5 cases while V in 17 cases. FS is the winning algorithm in only one case, and SMT has no wins. Although EPRL holds the second best rank, it has a satisfactory number of wins which shows its strength.

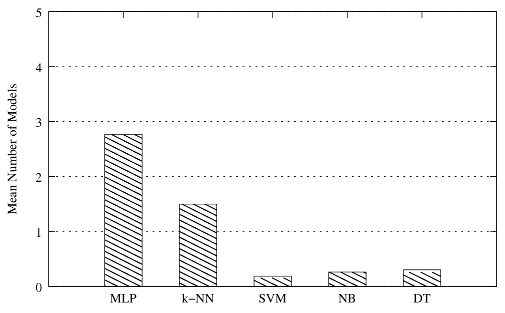
Figure 5 presents graphically the Nemenyi test with 90% confidence, $q_{0.10} = 2.291$ and $CD = 0.935$. We notice that there are two groups of similar algorithms. More specifically, the V algorithm is significantly better than FS and SMT, while both EPRL and FS have significantly better performance than SMT.

Table 5 presents the size of the final pruned ensemble for each algorithm on each dataset. Like in heterogeneous case, we notice that EPRL and FS

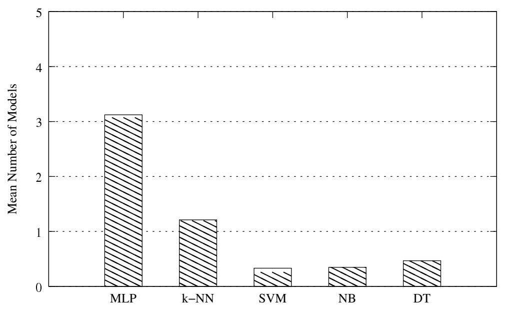
Table 3
Folder in UCI server and average size of the final ensemble for the heterogeneous case.

UCI Folder	FS	EPRL	SF
audiology	3.9	3.5	15.6
breast-cancer	3.4	6.7	100.0
breast-w	2.5	3.1	64.8
cmc	11.1	8.6	75.6
dermatology	2.9	1.0	45.5
ecoli	4.1	3.2	57.5
kr-vs-kp	4.2	3.7	42.8
glass	5.0	6.9	79.6
heart-h	2.1	5.2	96.9
hepatitis	1.5	1.9	100.0
image	14.6	9.8	37.0
ionosphere	1.9	3.4	51.0
iris	1.0	1.0	66.6
labor	1.0	1.0	100.0
lymph	2.1	3.8	97.0
diabetes	9.4	10.1	95.7
credit-a	7.1	10.6	71.1
credit-g	9.2	10.4	100.0
heart-statlog	9.3	6.2	74.4
heart-s	3.7	9.3	100.0
Average	5.0	5.47	73.55

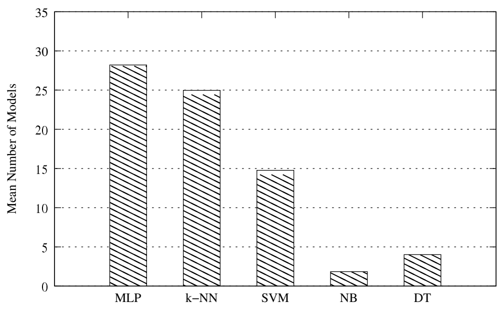
produce ensembles with an average of 7.94 and 5.67 models and thus with low computational cost. In general, EPRL manages to produce ensembles with high predictive performance and to decrease substantially the size of the full ensemble. This means that when the objective is to make an accurate decision and keep the computational and memory cost low, EPRL attains a great primacy against FS and the combination methods V and SMT.



(a) FS



(b) EPRL



(c) SF

Fig. 4. Type of selected models for each algorithm.

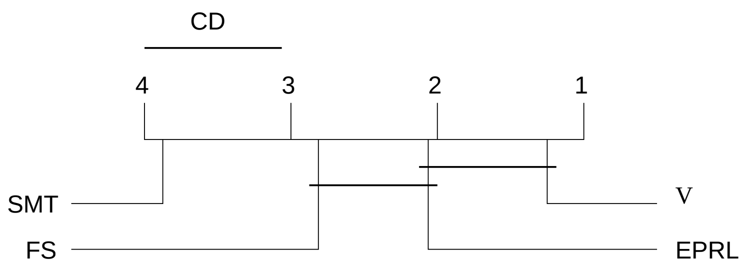


Fig. 5. Graphical representation of the Nemenyi test for the Homogeneous case.

6.3 Running times

Table 6 presents the running times of the different algorithms for one indicative dataset (image, 2310 instances and 7 classes). EPRL has the largest running time, 5.35 minutes, and follows SMT, 0.48 minutes, while FS and SF need only 0.21 and 0.16 minute respectively, to prune the ensemble.

7 Anytime Pruning

An interesting aspect of the proposed approach is its *anytime* property, which means that it can output a solution at any given time point. As we show from the experimental results the approach gradually improves by exploiting

Table 4
Folder in UCI server, accuracy and rank of each method on each of the 20 datasets for the homogeneous case.

	Accuracy				Rank			
UCI Folder	FS	EPRL	V	SMT	FS	EPRL	V	SMT
audiology	74.7±6.9	**76.2**±3.7	76.1±4.0	60.2±7.8	3.0	1.0	2.0	4.0
breast-cancer	73.9±4.3	73.9±4.8	**76.0**±4.1	62.6±4.5	2.5	2.5	1.0	4.0
breast-w	95.5±1.6	95.7±1.5	**95.8**±1.1	95.3±1.4	3.0	2.0	1.0	4.0
cmc	52.4±3.1	52.8±2.5	**53.8**±2.3	44.3±4.6	3.0	2.0	1.0	4.0
dermatology	94.2±2.1	94.4±2.8	**96.3**±3.3	91.9±2.7	3.0	2.0	1.0	4.0
ecoli	83.6±3.7	**85.1**±2.4	84.9±2.3	79.7±4.0	3.0	1.0	2.0	4.0
kr-vs-kp	**99.2**±0.3	**99.2**±0.3	**99.2**±0.3	98.8±0.3	2.0	2.0	2.0	4.0
glass	68.6±6.3	67.1±6.0	**71.0**±6.8	54.3±6.4	2.0	3.0	1.0	4.0
heart-h	77.6±3.7	**78.4**±3.2	77.9±3.1	75.5±4.2	3.0	1.0	2.0	4.0
hepatitis	78.4±5.5	78.4±7.9	**79.4**±5.2	77.4±5.7	2.5	2.5	1.0	4.0
image	96.4±0.6	**96.8**±0.8	**96.8**±0.7	94.1±1.1	3.0	1.5	1.5	4.0
ionosphere	90.6±2.2	90.6±2.6	**93.0**±2.4	86.1±3.2	2.5	2.5	1.0	4.0
iris	94.3±3.0	94.0±2.9	**96.3**±3.1	94.7±4.2	3.0	4.0	1.0	2.0
labor	72.7±1.1	74.5±1.2	**79.1**±1.0	54.5±1.1	3.0	2.0	1.0	4.0
lymph	75.2±7.0	77.6±9.0	**78.3**±9.0	65.9±8.2	3.0	2.0	1.0	4.0
diabetes	74.5±3.9	75.0±4.2	**75.3**±4.2	67.9±4.4	3.0	2.0	1.0	4.0
credit-a	86.7±2.1	86.9±2.2	**87.3**±2.3	83.8±3.1	3.0	2.0	1.0	4.0
credit-g	73.3±2.2	73.6±2.4	**75.2**±2.3	67.7±3.1	3.0	2.0	1.0	4.0
heart-statlog	77.2±5.9	80.0±5.4	**81.5**±3.7	71.9±4.3	3.0	2.0	1.0	4.0
heart-s	35.8±7.7	41.3±8.0	**42.9**±4.9	35.0±8.5	3.0	2.0	1.0	4.0
Average	78.7	79.6	80.8	73.1	2.825	2.05	1.225	3.9

the knowledge it acquired. However, as the ϵ parameter of the ϵ-greedy action selection method becomes very small, the exploration practically ceases and the agent fully exploits the acquired knowledge. As a consequence the performance converges with no further improvement. It would be very useful if the proposed approach continued improving with time, as the are many domains where predictive performance is critical and the data analyst is willing to sacrifice extra learning time in order to improve performance.

Table 5
Folder in UCI server and average size of the final ensemble for the homogeneous case.

UCI Folder	FS	EPRL
audiology	4.8	5.5
breast-cancer	4.4	7.6
breast-w	4.5	9.1
cmc	13.1	19.7
dermatology	2.4	4.9
ecoli	5.5	10.9
kr-vs-kp	2.5	3.6
glass	5.9	8.2
heart-h	4.9	4.1
hepatitis	2.5	3.7
image	9.7	8.0
ionosphere	3.4	5.3
iris	1.1	4.1
labor	1.5	1.7
lymph	3.5	5.8
diabetes	10.1	11.4
credit-a	5.7	8.7
credit-g	14.7	14.7
heart-statlog	6.7	9.5
heart-s	6.5	12.3
Average	5.67	7.94

Table 6
Running times of the algorithms for one indicative dataset.

FS	EPRL	SF	SMT
0.21 m	5.35 m	0.16 m	0.48 m

To accommodate this, we introduce the notion of learning periods, where a learning period consists of a number of learning episodes. When a learning period starts, the ϵ parameter is set to a high value, in order to explore the state space, and is decayed over episodes. A learning period ends when ϵ is less than a small threshold which indicates that the agent does not choose exploratory actions. At the same time, the weights of the network that approximates the action-value function are retained for the next learning period, so that the acquired knowledge is not forgotten.

To evaluate this extension of the proposed approach, we setup an experiment for both heterogeneous and homogeneous models. More specifically, the value of ϵ is set to 0.6 when a period finishes, and decays by a factor of 10^{-4}. A learning period ends when $\epsilon < 0.05$.

Table 7 depicts the average rank of the competing algorithms across all datasets for the 4 first periods in the heterogeneous case. Note that only the absolute performance of EPRL changes with respect to the periods, leading to a change of its average rank whenever its performance exceeds that of a previously better method. However, as ranks are a relative measure of performance, the rank of all methods changes as a result of the change in the absolute performance of EPRL. We notice that EPRL gradually improves its performance per period and it finally outperforms the other algorithms and especially FS and SF. This evidence indicates that the anytime pruning extension improves the performance of EPRL over time.

Table 7
Average rank of all algorithms for the heterogeneous case.

Period	FS	EPRL	SF	V	SMT
1	2.775	2.625	2.5	3.8	3.725
2	2.725	2.225	2.55	3.8	3.775
3	2.8	1.95	2.7	3.85	3.775
4	2.85	1.8	2.75	3.85	3.825

Table 8 shows the average rank of the algorithms for the different number of periods. As it can be seen, only in the second period EPRL improves slightly its overall performance while in the next two periods there are no changes in the average ranks.

8 Conclusions

This paper has presented a method for pruning an ensemble of classifiers based on Reinforcement Learning. The proposed method evaluated on a large number of datasets and compared with other pruning methods, as well as with a state-of-the-art combination method. The experiment carried out on

Table 8
Average rank of all algorithms for the homogeneous case.

Period	FS	EPRL	V	SMT
1	2.7	2.025	1.15	3.9
2	2.8	1.875	1.3	3.95
3	2.8	1.875	1.3	3.95
4	2.8	1.875	1.3	3.95

both Heterogeneous and Homogeneous ensembles using an initial pool of 100 models.

The results have shown that the proposed approach obtains high predictive performance, especially in the Heterogeneous case. Additionally, EPRL produces small sized ensembles and thus reduces the needed computational and memory resources. The proposed approach is general and can be used for optimizing any specific performance evaluation metric and with any ensemble combination method. In addition it can output a solution anytime and it has the ability of improving over time, which makes it suitable for domains where time can be sacrificed in order to improve performance.

Another interesting property is that the computational complexity of the method is linear with respect to the ensemble size, as each training episode lasts as many time steps as the number of classifiers in the ensemble. However, the state space that the agent has to explore grows exponentially with the number of classifiers, and so does the complexity of the learning problem. This of course is a problem for any search-based pruning method.

References

[1] D. N. A. Asuncion, UCI machine learning repository (2007).
URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`

[2] 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.

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

[4] R. Caruana, A. Niculescu-Mizil, G. Crew, A. Ksikes, Ensemble selection from libraries of models, in: Proceedings of the 21st International Conference on Machine learning, ICML' 04, Banff, Alberta, Canada, 2004.

[5] J. Demsar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1–30.

[6] T. G. Dietterich, Machine-learning research: Four current directions, The AI Magazine 18 (4) (1998) 97–136.

[7] T. G. Dietterich, Ensemble methods in machine learning, in: Proceedings of the 1st International Workshop on Multiple Classifier Systems, MCS '00, 2000.

[8] C. Dimitrakakis, S. Bengio, Online adaptive policies for ensemble classifiers, Trends in Neurocomputing 64 (2005) 211–221.

[9] S. Dzeroski, B. Zenko, Is combining classifiers with stacking better than selecting the best one?, Machine Learning 54 (3) (2004) 255–273.

[10] W. Fan, F. Chu, H. Wang, P. S. Yu, Pruning and dynamic scheduling of cost-sensitive ensembles, in: Proceedings of the 19th National Conference on Artificial intelligence, Edmonton, Alberta, Canada, 2002.

[11] M. Friedman, A comparison of alternative tests of significance for the problem of m rankings, Annals of Mathematical Statistics 11 (1940) 86–92.

[12] G. Giacinto, F. Roli, An approach to the automatic design of multiple classifier systems, Pattern Recognition Letters 22 (1) (2001) 25–33.

[13] R. L. Iman, J. M. Davenport, Approximations of the critical region of the friedman statistic, Communications in Statistics (1980) 571–595.

[14] L. P. Kaelbling, M. L. Littman, A. P. Moore, Reinforcement learning: A survey, Journal of Artificial Intelligence Research 4 (1996) 237–285.

[15] M. G. Lagoudakis, M. L. Littman, Algorithm selection using reinforcement learning, in: Proceedings of the 17th International Conference on Machine Learning, 2000.

[16] A. Lazarevic, Z. Obradovic, Effective pruning of neural network classifiers, in: 2001 IEEE/INNS International Conference on Neural Networks, IJCNN 2001, 2001.

[17] D. Margineantu, T. Dietterich, Pruning adaptive boosting, in: Proceedings of the 14th International Conference on Machine Learning, 1997.

[18] G. Martinez-Munoz, A. Suarez, Aggregation ordering in bagging, in: International Conference on Artificial Intelligence and Applications (IASTED), Acta Press, 2004.

[19] G. Martinez-Munoz, A. Suarez, Pruning in ordered bagging ensembles, in: 23rd International Conference in Machine Learning (ICML-2006), ACM Press, 2006.

[20] G. Martinez-Munoz, A. Suarez, Using boosting to prune bagging ensembles, Pattern Recognition Letters 28 (1) (2007) 156–165.

[21] P. B. Nemenyi, Distribution-free multiple comparisons, Ph.D. thesis, Princeton University (1963).

[22] I. Partalas, G. Tsoumakas, I. Katakis, I. Vlahavas, Ensemble pruning using reinforcement learning, in: Proceedings of the 4th Hellenic Conference on Artificial Intelligence (SETN 2006), Heraklion, Greece, 2006.

[23] A. Prodromidis, P. Chan, Meta-learning in distributed data mining systems: Issues and approaches, in: H. Kargupta, P. Chan (eds.), Advances of Distributed Data Mining, MIT/AAAI Press, 2000.

[24] A. Prodromidis, S. J. Stolfo, Cost complexity-based pruning of ensemble classifiers, Knowledge and Information Systems 3 (4) (2001) 449–469.

[25] F. Provost, V. Kolluri, A survey of methods for scaling up inductive algorithms, Data Mining and Knowledge Discovery 3 (2) (1999) 131–169.

[26] R. E. Schapire, The strength of weak learnability, Machine Learning 5 (1990) 197–227.

[27] W. N. Street, Y. Kim, A streaming ensemble algorithm (sea) for large-scale classification, in: 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2001.

[28] R. S. Sutton, A. G. Barto, Reinforcement Learning, An Introduction, MIT Press, 1999.

[29] C. Tamon, J. Xiang, On the boosting pruning problem, in: 11th European Conference on Machine Learning (ECML 2000), Springer-Verlag, 2000.

[30] G. Tsoumakas, L. Angelis, I. Vlahavas, Selective fusion of heterogeneous classifiers, Intelligent Data Analysis 9 (6) (2005) 511–525.

[31] G. Tsoumakas, I. Katakis, I. Vlahavas, Effective voting of heterogeneous classifiers, in: Proceedings of the 15th European Conference on Machine Learning (ECML 2004), Pisa, Italy, 2004.

[32] G. Tsoumakas, I. Vlahavas, Distributed data mining of large classifier ensembles, in: 2nd Hellenic Conference on Artificial Intelligence, 2002.

[33] H. Wang, W. Fan, P. S. Yu, J. Han, Mining concept-drifting data streams using ensemble classifiers, in: 9th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, New York, NY, USA, 2003.

[34] C. Watkins, P. Dayan, Q-learning, Machine Learning 8 (1992) 279–292.

[35] I. H. Witten, E. Frank, Data Mining: Practical machine learning tools and techniques, 2nd Edition, Morgan Kaufmann, 2005.

[36] D. H. Wolpert, Stacked generalization, Neural Networks 5 (1992) 241–259.

[37] Y. Zhang, S. Burer, W. N. Street, Ensemble pruning via semi-definite programming, Journal of Machine Learning Research 7 (2006) 1315–1338.

[38] Z. Zhou, W. Tang, Selective ensemble of decision trees, in: Proceedings of the 9th International Conference on Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing, RSFDGrC 2003, Chongqing, China, 2003.