

Ensemble Pruning using Reinforcement Learning

Ioannis Partalas, Grigorios Tsoumakas, Ioannis Katakis and Ioannis Vlahavas

Department of Informatics,
Aristotle University of Thessaloniki,
54124 Thessaloniki, Greece
{partalas,greg,katak,vlahavas}@csd.auth.gr

Abstract. Multiple Classifier systems have been developed in order to improve classification accuracy using methodologies for effective classifier combination. Classical approaches use heuristics, statistical tests, or a meta-learning level in order to find out the optimal combination function. We study this problem from a Reinforcement Learning perspective. In our modeling, an agent tries to learn the best policy for selecting classifiers by exploring a state space and considering a future cumulative reward from the environment. We evaluate our approach by comparing with state-of-the-art combination methods and obtain very promising results.

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 [1]. The success of this area is due to the fact that ensembles of predictive models offer higher predictive accuracy than individual models.

The first phase of an Ensemble Method is 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 [2]. Models that derive from running different learning algorithms on the same data set are called Heterogeneous. The second phase of an Ensemble Method is the combination of the models. Common methods here include Selection, Voting, Weighted Voting and Stacking. Recent work [3], [4] has shown that an additional intermediate phase of pruning an ensemble of heterogeneous classifiers (and combining the selected classifiers with Voting) leads to increased predictive performance.

This paper presents a Reinforcement Learning approach to the problem of pruning an ensemble of heterogeneous classifiers. We use the Q-learning algorithm in order to approximate an optimal policy of choosing whether to include or exclude each algorithm from the ensemble. We compare our approach against other pruning methods and state-of-the-art Ensemble methods and obtain very

promising results. Additionally, the proposed approach is an anytime algorithm, meaning that we can obtain a solution at any point.

The rest of this paper is structured as follows: Section 2 presents background information on reinforcement learning and heterogeneous classifier combination, focusing on material that will be later on referred in the paper. Section 3 reviews related work on pruning ensembles of heterogeneous models, as well as on using reinforcement learning for the combination of different algorithms. Section 4 presents our approach and Section 5 the setup of the experiments for its evaluation. Section 6 discusses the results of the experiments and finally Section 7 concludes this work and points to future research directions.

2 Background

2.1 Reinforcement Learning

Reinforcement Learning (RL) addresses the problem of how an agent can learn a behavior through trial-and-error interactions with a dynamic environment [5]. In an RL task the agent, at each time step, 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, $r_{t+1} \in \mathfrak{R}$, and moves to a new state s_{t+1} . The objective of the agent is to maximize the cumulative reward received over time. More specifically, the agent selects actions that maximize the expected discounted return:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \quad (1)$$

where γ , $0 \leq \gamma < 1$, is the discount factor and expresses the importance of future rewards.

A *policy* π specifies that in state s the probability of taking action a is $\pi(s, a)$. For any policy π , the value of state s , $V^\pi(s)$, 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:

$$V^\pi(s) = E_\pi \{R_t \mid s_t = s\} = E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s \right\}, \quad (2)$$

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 :

$$Q^\pi(s, a) = E_\pi \{R_t \mid s_t = s, a_t = a\} = E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a \right\}. \quad (3)$$

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^* :

$$Q^*(s, a) = E \left\{ r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a') \mid s_t = s, a_t = a \right\} \quad (4)$$

The optimal policy can now be defined as:

$$\pi^* = \arg \max_a Q^*(s, a) \quad (5)$$

The most widely used algorithm for finding the optimal policy is the Q-learning algorithm [6] which approximates the Q function with the following form:

$$Q(s_t, a_t) = r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a'). \quad (6)$$

2.2 Combining Heterogeneous Classification Models

A lot of different ideas and methodologies have been proposed in the past for the combination of heterogeneous classification models. The main motivation behind this research is the common observation that there is no independent 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.

Another simple method is Evaluation and Selection. This method evaluates each of the models (typically using 10-fold cross-validation) on the training set and selects the best one for application to the test set.

Stacked Generalization [7], 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 $D_1 \dots D_k$ of equal size. For each fold $i = 1 \dots k$ 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.

3 Related Work

3.1 Pruning Ensembles of Classifiers

Most of the Ensemble Methods in the literature deal either with the production or the combination of multiple classifiers. However, recent work has shown that pruning an ensemble of classifiers (and combining the selected classifiers with Voting) leads to increased predictive performance.

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 via forward stepwise selection of the classification models. As a heuristic, they use the accuracy of combining the selected classifiers with the method of voting. This way they manage to achieve very good predictive performance compared to state-of-the-art ensemble methods.

In [3], pruning is performed using statistical procedures that determine whether the differences in predictive performance among the classifiers of the ensemble are significant. Using such procedures 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.

3.2 Reinforcement Learning for Algorithm Combination

Research work on utilizing Reinforcement Learning (RL) for algorithm combination is limited. We found two past approaches on this subject, one applied to the problem of selecting a single classification algorithm and one of applying the most suitable algorithms on different segments of the dataset. The latest approach applied on two computational problems: order statistic selection and sorting.

In [8], RL is used 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.

In [9], the problem of algorithm selection is formulated as a Markov Decision Process (MDP) and an RL approach is used 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 current instantiation of the instance features 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 Ensemble Pruning via Reinforcement Learning

First we must formulate the problem of pruning an ensemble of classifiers as an RL task. To do that, we must define the following components:

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

In our approach, a state represents the set of classifiers that have been selected so far and thus S is the powerset of S_c , $S = P(S_c)$, where $S_c = \{C_1, \dots, C_n\}$ is the set of classifiers. S has 2^n different states, where n is the number of classifiers available for selection. In each state the agent can select between two actions. It can either include an algorithm into the ensemble or not and thus $A = \{include(C_i) | i = 1 \dots n\} \cup \{exclude(C_i) | i = 1 \dots n\}$. Finally, the reward is the accuracy that we obtain if we combine the selected classifiers with the method of voting.

The training phase consists of running a number of episodes, where an *episode* is defined as a sequence of agent-environment interactions. In our approach, each episode starts with an empty set of classifiers and lasts n time steps. At each time step, $t = 1 \dots n$, of the episode, the agent chooses to include or not a specific classifier into the ensemble, $A(s_{t-1}) = \{include(C_t), exclude(C_t)\}$. Subsequently, the agent receives an immediate reward which equals to the accuracy of the current subset of classifiers combined with voting. The update equation of Q-learning is:

$$Q(s_t, a_t) = Accuracy(s_{t+1}) + \gamma \max_{a'} Q(s_{t+1}, a') \quad (7)$$

The episode ends when the decision to include or exclude the n^{th} algorithm is taken. Figure 1 graphically shows a training episode.

During the training phase 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 softmax action selection method, where an action a is selected with probability:

$$P(a) = \frac{\exp^{Q(s,a)/T}}{\sum_{a'} \exp^{Q(s,a')/T}}, \quad (8)$$

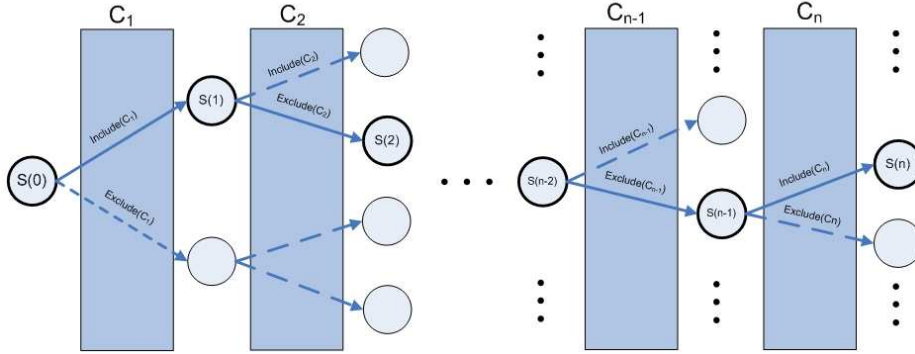


Fig. 1. The procedure of selecting a sequence of classifiers.

where T is a positive parameter, called *temperature*, which starts from a high value and gradually is reduced until it becomes zero. High temperature values assign equal probabilities to all actions so the agent explores the state space, while in other case low temperatures cause high probabilities for action with high value estimates and so the agent exploits his knowledge.

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. This way, the problem of pruning an ensemble of n classifiers has been transformed into the reinforcement learning task of letting an agent learn an optimal policy of taking n actions to maximize the cumulative reward.

5 Experimental Setup

We compare the performance of our approach, Ensemble Pruning via Reinforcement Learning (EPRL), against the following state-of-the-art classifier combination methods: Stacking with Multi-Response Model Trees (SMT), Evaluation and Selection (ES) and Effective Voting (EV).

The methods are applied on top of a heterogeneous ensemble produced using the WEKA [10] implementations of the following 9 different classification algorithms:

- DT: the decision table algorithm of Kohavi [11].
- JRip: the RIPPER rule learning algorithm [12].
- PART: the PART rule learning algorithm [13].
- J48: the decision tree learning algorithm C4.5 [14], using Laplace smoothing for predicted probabilities.
- IBk: the k nearest neighbor algorithm [15].
- K^* : an instance based learning algorithm with entropic distance measure [16].

- NB: the Naive Bayes algorithm using the kernel density estimator rather than assume normal distributions for numeric attributes [17].
- RBF: WEKA implementation of an algorithm for training a radial basis function network [18].
- MLP: WEKA implementation of an algorithm for training a multilayer perceptron [18].

We compare the methods on 11 data sets from the UCI Machine Learning repository [19]. 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).

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
hepatitis	155	2	6	13	5.67
heart-disease (cleveland)	303	5	6	7	0.18
horse-colic	368	2	7	15	23.80
iris	150	3	4	0	0.0
labor	57	2	8	8	35.75
ionosphere	351	2	34	0	0.0
prima-indians-diabetes	768	2	8	0	0.00
soybean	683	19	0	35	9.78
voting-records	435	2	0	16	5.63
wine	178	7	1	16	0.00
zoo	101	7	1	16	0.00

For the evaluation of the methods we perform a 10-fold stratified cross-validation experiment. In each of the 10 repetitions, the same 9 folds are used for training the different methods and 1 fold for evaluating their performance. The accuracy rates are averaged over the 10 folds in order to obtain the average accuracy $acc_m(d_i)$ of each method m in each data set d_i .

Our approach is run using a fixed number of 2000 episodes. For the representation of the value function Q we used a tabular approach, where each value $Q(s, a)$ is stored in a table. The value of T was set to 1000 and decreased by a factor of 2% at each episode. The decisions of the classifiers in the final pruned ensemble are combined using Voting.

6 Results and Discussion

Table 3 presents the accuracy of each classifier combination method on each of the 11 data sets. The last row presents the geometric mean of the accuracy over all data sets. The highest accuracy for each data set is emphasized using bold typeface.

Table 2. Folder in UCI server, average accuracy of each combining method on each of the 11 data sets and geometric mean of each combining method over all data sets

UCI Folder	SMT	ES	EV	EPRL
hepatitis	0.8379	0.8383	0.8383	0.8614
heart-disease (cleveland)	0.8300	0.8442	0.8172	0.8170
horse-colic	0.8290	0.8452	0.8535	0.8481
iris	0.9533	0.9533	0.9467	0.9533
labor	0.9100	0.9433	0.9267	0.9667
ionosphere	0.9402	0.9147	0.9232	0.9203
prima-indians-diabetes	0.7603	0.7720	0.7681	0.7759
soybean	0.9254	0.9254	0.9444	0.9415
voting-records	0.9586	0.9518	0.9563	0.9542
wine	0.9663	0.9722	0.9889	0.9722
zoo	0.9509	0.9409	0.9609	0.9609
geometric mean	0.8940	0.8979	0.8996	0.9040

We first notice that our approach has the highest mean accuracy than the rest of the methods. In addition, it has the highest accuracy in 5 data sets, one more than Effective Voting and two more than Stacking with Multi-Response Model Trees that are considered to be state-of-the-art methods for the combination of heterogeneous classifiers. Figure 2 shows the average accuracy of our approach over all datasets using a varying number of episodes, starting from 250 up to 2000 with a step of 250 episodes. It also shows the accuracy of the other classifier combination methods for comparison purposes. As it can be seen, our approach obtains good performance for a small number of episodes and finally outperforms the other methods. In approximately 500 episodes T has acquired a significantly low value and the agent exploits its knowledge. As a result, the accuracy of our approach increases rapidly.

7 Conclusions and Future Work

This paper has presented a method for pruning an ensemble of heterogeneous classifiers based on Reinforcement Learning. The results of combining the subset of classifiers with Voting are very promising as they compare favorably against the results of state-of-the-art methods for heterogeneous classifier combination.

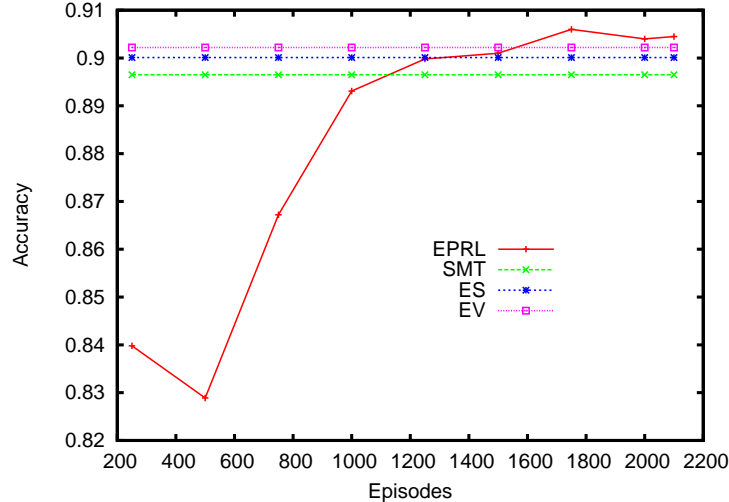


Fig. 2. Average accuracy on all data sets with respect to the episodes.

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, after an initial period of exploration, the approach starts improving by exploiting the knowledge it acquired. Therefore the more the training episodes, the higher the predictive performance of the resulting pruned ensemble, until the approach converges around some good 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.

In the future, we intend to investigate the applicability of the proposed ideas in libraries of thousands of models [4]. In such a case we need to alter the representation of the states, in order to tackle the explosion of the state space. In addition, we plan to explore other methods of action selection, in order to improve not only the exploration of the state space but also the time needed for the algorithm to converge.

Acknowledgements

This work was partially supported by the Greek R&D General Secretariat through a PENED program (EPAN M.8.3.1, No. 03E Δ 73).

References

1. Dietterich, T.G.: Machine-learning research: Four current directions. *The AI Magazine* **18**(4) (1998) 97–136
2. Dietterich, T.G.: Ensemble methods in machine learning. *Lecture Notes in Computer Science* **1857** (2000) 1–15
3. Tsoumakas, G., Katakis, I., Vlahavas, I.P.: Effective voting of heterogeneous classifiers. In: *Proceedings of the 15th European Conference on Machine Learning, ECML 04.* (2004) 465–476
4. Caruana, R., Niculescu-Mizil, A., Crew, G., Ksikes, A.: Ensemble selection from libraries of models. In: *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, New York, NY, USA, ACM Press (2004) 18
5. Sutton, R.S., Barto, A.G.: *Reinforcement Learning, An Introduction*. MIT Press (1999)
6. Watkins, C., Dayan, P.: Q-learning. *Machine Learning* **8** (1992) 279–292
7. Wolpert, D.H.: Stacked generalization. Technical Report LA-UR-90-3460, Los Alamos, NM (1990)
8. Christos Dimitrakakis, S.B.: Online adaptive policies for ensemble classifiers. *Trends in Neurocomputing* **64** (2005) 211–221
9. Lagoudakis, M.G., Littman, M.L.: Algorithm selection using reinforcement learning. In: *Proc. 17th International Conf. on Machine Learning*, Morgan Kaufmann, San Francisco, CA (2000) 511–518
10. Witten, I.H., Frank, E.: *Data Mining: Practical machine learning tools and techniques*, 2nd Edition. Morgan Kaufmann (2005)
11. Kohavi, R.: The power of decision tables. In Lavrac, N., Wrobel, S., eds.: *Proceedings of the European Conference on Machine Learning*. *Lecture Notes in Artificial Intelligence* 914, Berlin, Heidelberg, New York, Springer Verlag (1995) 174–189
12. Cohen, W.W.: Fast effective rule induction. In Prieditis, A., Russell, S., eds.: *Proc. of the 12th International Conference on Machine Learning*, Tahoe City, CA, Morgan Kaufmann (1995) 115–123
13. Frank, E., Witten, I.H.: Generating accurate rule sets without global optimization. In: *Proc. 15th International Conf. on Machine Learning*, Morgan Kaufmann, San Francisco, CA (1998) 144–151
14. Quinlan, J.R.: *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1993)
15. Aha, D.W., Kibler, D., Albert, M.K.: Instance-based learning algorithms. *Mach. Learn.* **6**(1) (1991) 37–66
16. Cleary, J.G., Trigg, L.E.: K*: an instance-based learner using an entropic distance measure. In: *Proc. 12th International Conference on Machine Learning*, Morgan Kaufmann (1995) 108–114
17. John, G.H., Langley, P.: Estimating continuous distributions in Bayesian classifiers. In: *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*. (1995) 338–345
18. Bishop, C.M.: *Neural networks for pattern recognition*. Oxford University Press, Oxford, UK (1996)
19. D.J. Newman, S. Hettich, C.B., Merz, C.: *UCI repository of machine learning databases* (1998)