Instance-Based Ensemble Selection Using Deep Reinforcement Learning

Zhengshang Liu
School of Computing and Information Systems
The University of Melbourne
Parkville, Australia
zhengshangl@student.unimelb.edu.au

Kotagiri Ramamohanarao
School of Computing and Information Systems
The University of Melbourne
Parkville, Australia
kotagiri@unimelb.edu.au

***Abstract*—Ensemble selection is a very active research topic in machine learning area. It aims to achieve a better performance by selecting a proper subset of the original ensemble, which is essentially a searching problem in large combinatorial spaces. In this paper, we propose an instance-based reinforcement learning (IBRL) model, that selects distinct subsets for different instances. Specifically, we use deep Q-network to approximate the optimal policy. Rather than considering the overall performance of each classifier, the network learns from the feedback of classifiers on individual instance, so that it generates non-static subsets for different instances. Experiments are conducted to compare our model against state-of-the-art approaches for both selection and combination. The proposed method generates promising results and it shows exceptional advantage in large scale distributed environment. Due to the environment-free characteristic of reinforcement learning, our model is adaptable to various real world tasks with minimal changes.**

***Index Terms*—Ensemble Selection, Reinforcement Learning, Instance-based**

I. Introduction

Ensemble method is a very active research topic in machine learning area. It refers to construct proper combination of classifiers [1]. The fundamental principle behind is when there is no correlation between errors made by different classifiers, the aggregated result tends to be more accurate [2]. Consequently, ensemble methods have been applied to improve the performances of individual classification or regression models.

Originally, ensemble methods consist of two major steps: sub-model construction and combination. Some recent studies [3], [4], [5], [6], [7] have introduced an extra step, termed as ensemble selection or ensemble pruning. The main purpose of this step is to pursue a higher accuracy by finding the subset of most promising classifiers, rather than considering all of them through complex weighting functions.

Selecting the optimal subset from an ensemble of classifiers is an NP-complete problem [8]. Some studies [9], [10] apply greedy approaches to reduce the searching space. However, these approaches may lead to a suboptimal subset since they only visit a small number of combinations. Reinforcement learning, especially with approximation methods, tackles problems that have large state spaces [11]. Research [6] performs ensemble pruning using reinforcement learning, namely Q-learning with eligibility trace.

The ensemble selection problem is essentially a searching problem with a large amount of possible answers. All existing solutions generate a fixed set of classifiers for all instances. In this paper, we introduce an instance-based reinforcement learning (IBRL) model which produces different sets of classifiers for different instances. Figure 1 shows a hypothetical distribution of data, such that data instances in a certain group can be perfectly predicted by a set of classifiers. Once we find such distribution and corresponding classifiers, we have the optimal solution to the ensemble selection problem. The proposed method applies reinforcement learning to gain a strong comprehension over the entire state space, and it produces instance-based predictive results. When evaluating a classifier, our model refers not only to the overall error rate, but also the performance on a specific label.

In section II, we cover background and related works. Our proposed model is illustrated in section III, experimental results and analysis are shown in section IV. Our main contributions in this paper are 1) we propose a novel model that achieves instance-based behaviour; 2) our approach provides higher accuracy on prediction tasks, and shows significant advantage in distributed environment; 3) we study how homogeneous and heterogeneous ensembles benefit different ensemble methods.

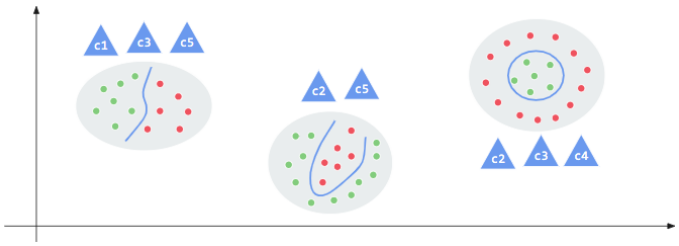

Fig. 1. Hypothetical distribution of data

II. Background and Related Work

A. Ensemble Methods

1) Ensemble Construction: Training sub-models with identical datasets usually leads to common flaws. This problem can be addressed by manually injecting variance into sub-models.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

A well-known methodology is bootstrapping (sometimes referred as bagging) [12]. Bootstrapping can also be applied on features: the random forests [13] algorithm splits its trees, at each node, using a group of random selected features.

An ensemble can be composed by homogeneous and heterogeneous sub-models. Different models that follow different formulations have different views and assumptions on the data. For example, comparing to k-NN, multilayer perceptrons are more robust to noise. Recent research [14] shows that ensembles of hybrid classifiers have significantly smaller error rates.

2) Model Aggregation: Two most common ways to combine an ensemble of models are majority vote and weighted vote. In a majority vote schema, each classifier outputs a predicted label. The label that proposed by most classifiers is then chosen to be the final result. On the other hand, when using weighted vote, classifiers produce probabilistic measurements rather than deterministic judgements. Those measurements are then summed up with respect to each label, and the label with highest cumulative probability is regarded as the final output.

A greedy approach, forward selection (FS) was proposed in [10], which constructs an ensemble in a hillclimbing style. Caruana et al. [9] apply a similar method to prune an ensemble of 1000 heterogeneous classifiers that are trained by different sets of parameters. The procedure of this method is straightforward: 1) it starts with an empty ensemble, 2) it adds a new classifier to the ensemble as long as it contributes to a higher accuracy on the validation set, 3) it stops when all classifiers are selected or no further improvement can be made.

Stacking, sometimes named stacked generalization, is another way to combine multiple classifiers [3], [7]. Stacking method consists of two parts: base-level (level 0) models and meta-level (level 1) model, and it aims to learn a meta-algorithms to combine those base models.

B. Reinforcement Learning

Reinforcement learning emphasizes on finding the optimal way to interact with an environment even when the mechanism behind is unclear to the agent [11]. Most reinforcement learning problems are modeled as a finite Markov decision process (MDP), which consists of four elements: states, actions, state transition probability function, and reward function. The agent is guided by a policy $\pi(a \mid s)$ which maps actions to values given a state. Our objective is to find a policy that maximizes the accumulated rewards. Given a policy, we can define the Q-function of a state-action pair as the synthesis of sequential rewards when starting from that certain state and action:

$$Q_\pi(s,a) = E_\pi[\sum_{t=0}^{\infty} \gamma^t R_t \mid S_0 = s, A_0 = a]$$

Q-learning [15] is a common approach to utilize Q-function and solve the reinforcement learning problem. It updates the Q-value through an iterative process:

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma max_{a'} Q(s',a') - Q(s,a))$$

where α is the learning rate, γ is the discount factor, and s' is the resulting state by applying action a in state s.

C. Deep Q Network

In many reinforcement learning tasks, the Q-function is approximated using a linear combination of some features. However, simple linear combination is not accurate enough for problems with large state spaces. As a versatile structure, deep neural network is one of the best candidate to fill this gap.

Mnih et. al [16] first used a deep neural network to approximate Q-function and proposed the idea of deep Q-network (DQN). The neural network takes the state (e.g. image in the original research) as input, and outputs the Q-value for all possible actions. Two approaches, experience replay and delayed update, are applied to make the training more stable.

D. Ensemble Pruning via RL

A related research that prunes an ensemble via reinforcement learning (EPRL) was conducted recently [6]. In order to map an ensemble selection problem into reinforcement learning model, the authors define the problem as follows:

1) $C = c_1, c_2, ..., c_n$ is a set of classifiers to be pruned;
2) A state is a tuple (C', c_i), where C' is the current ensemble and c_i is the classifier that currently under consideration;
3) There are two available actions, $include(c_i)$ and $exclude(c_i)$, for any given state except the very last one;
4) The reward is zero everywhere except for the final evaluation action, and that reward is correlated to the predictive performance of the ensemble.

Similar to other ensemble methods, this model generates a static subset of ensemble for all validation instances. This solution leads to two disadvantages: 1) a particular subset of ensemble could be the optimal choice for some instances but suboptimal for others; 2) in validation phases, all instances have to iterate through all classifiers, and consequently, large computational overheads are required.

Reinforcement learning is not widely used in ensemble selection (pruning). Besides this similar study, two related models are proposed. These studies provide us with possibilities to address ensemble selection problem with reinforcement learning method. Dimitrakakis and Bengio [17] apply reinforcement learning to select supervised learning classifiers. More precisely, this model consists of a group of multi-layer perceptrons (MLPs) that serve as classifiers, and one additional top-level MLP that takes training instance as input and outputs the classifier that fits the instance best. The selected classifier will then be trained using that instance. Lagoudakis and Littman [18] use Markov decision process (MDP) to simulate algorithm selection problem and solve with reinforcement learning, specifically a variation of the Q-learning algorithm. Their goal is to select an algorithm that can solve a given task best. The states of this MDP are the feature representations of tasks, and the actions are choosing different algorithms. There is only one state transition, so the

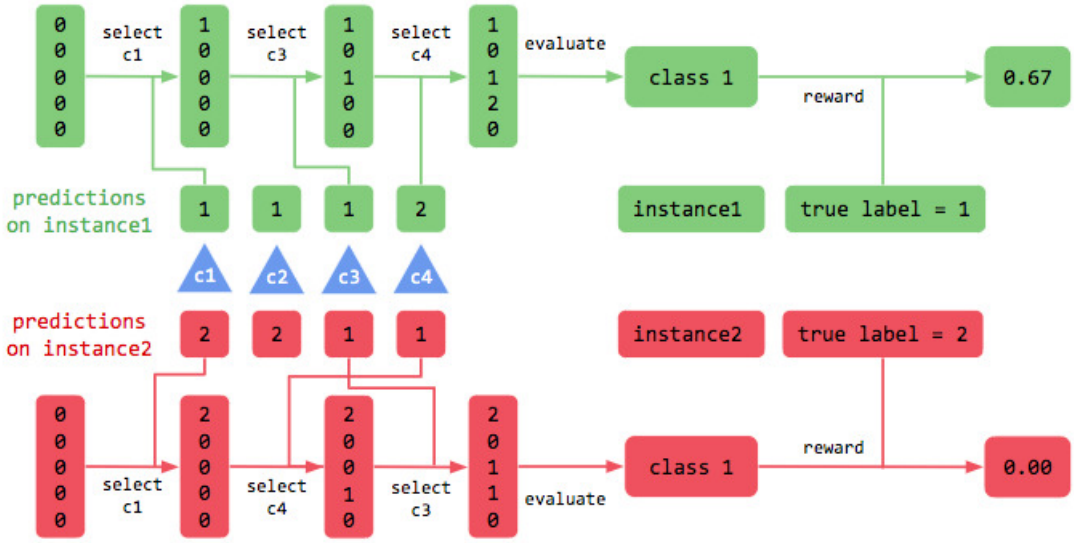

Fig. 2. Agent's activity in one episode

only reward is associated with algorithm selection, and it is relevant to the cost of running that algorithm.

III. Model

We formulate the ensemble selection problem into a reinforcement learning model, which contains the following components:

1) the training dataset with k instances, where $X = \{x_1, x_2, ..., x_k\}$ is the set of features and $Y = \{y_1, y_2, ..., y_k\}$ is the set of corresponding true labels;
2) an ensemble with n classifiers: $C = \{c_1, c_2, ..., c_n\}$;
3) the state s_i consists of the prediction results on the i-th instance. It is represented as a tuple $(c_1(x_i), c_2(x_i), ..., c_n(x_i))$, where $c_j(x_i)$ is the prediction from classifier c_j on data instance x_i. If a classifier has not been selected yet, then $c(x_i) = nil$. Consequently, the entire state space is $S = \{s_i \mid i \in [1, k]\}$;
4) two types of actions: visiting a classifier if it is unused, or evaluating the ensemble at current state if at least one classifier has been visited. Given a state s_i, the action space is

$$A(s_i) = \{v(c_j) \mid c_j(x_i) = nil, j \in [i, n]\} \cup \{e(s_i) \mid s_i \; is \; not \; empty\};$$

5) the reward function that covers three different scenarios:
   a) visiting any classifier gives no reward $R(v(c)) = 0$,
   b) incorrect evaluation also gives zero reward $R(e(s_i) \neq y_i) = 0$,
   c) if the evaluation produces correct result, the reward depends on the proportion of accurate classifiers $R(e(s_i) = y_i) = \frac{\#\, of\; accurate\; classifiers}{\#\, of\; selected\; classifiers}$;

   and
6) the policy function which takes a state as input and outputs the most desirable action.

Our objective is to find a policy which maximizes the overall rewards on the validation dataset.

The activity of our agent is illustrated in figure 2. For each episode, our agent starts with an empty state and ends with evaluation action. A validation instance is provided and it determines the reward and states transition within that episode. As shown in figure 2, $2/3$ of the ensemble correctly classify instance 1 (green), so the final reward is 0.67. On the other hand, the ensemble make mistake on instance 2 (red), which leads to zero reward. The number of steps that our agent performs in a single episode is bounded by the number of classifiers plus the evaluation, that is, $n + 1$. At each step, the agent selects an action using the following exploration strategy: with certain probability, it randomly selects an available action, otherwise, it selects the action with highest Q-value. After a classifier is selected, our agent retrieves the predictive result from that classifier and uses it to generate the next state. We utilize a fractional decreasing exploration rate, which has an initial value of 0.5 and decreases by 8×10^{-6} in each episode.

When applying nonlinear function such as deep neural network to approximate the Q-function, the reinforcement learning process becomes unstable or even diverge [16]. The fundamental reason for this problem is that reinforcement learning (especially Temporal-Difference method) relies on the self defined Bellman equation. The Q-function of next generation is heavily impacted by the current one in two ways. On one hand, the TD estimation depends fully on the existing Q-function. On the other hand, the reward from the environment is affected by our policy, and the policy also relies on the current Q-function. As a result, a slight change to the Q-function changes both the state-value evaluation and the policy distribution, and thus, updates the Q-function for the next iteration in an unpredictable way. In order to circumvent this problem, we use mini batch to train the neural network. When an action is performed, rather than updating

the target Q-network immediately, we memorize all related information of this state transition and save as a history tuple (s, a, s', r). Those collected history tuples are then split into two groups. The first group contains all tuples that are related to evaluation action. All records in this group will be considered when updating the Q-network since each record in this group involves a non-zero reward. The second group contains the rest tuples. Instead of selecting all records in this group, we sample a portion of it. Because no reward is involved in actions other than evaluation, so the target Q-value fully depends on the value of the next state, and it is meaningless to perform such update massively before our Q-network is trained preliminarily. We use a negative quadratic function to sample the training records dynamically:

$$s\_rate(t) = \max(\frac{-t^2 + \beta \times 10^5 t}{2\beta \times 10^{10}}, 0.5),$$

where t is the current episode, and β is a tunable parameter. We use 4 for $beta$ in this paper and the s_rate starts with 0 and reaches its pinnacle, 0.5, at the $200k$-th episode. After that, the sampling rate remains unchanged. Given the total training episode T, the time complexity of our training process is the integral of the s_rate function:

$$\int_0^T s\_rate(t) \times k \times n.$$

Q-learning is applied to solve our reinforcement learning model. Our policy function is updated at iteration i using the minimum square error between observed Q-value and approximated Q-value:

$$L_i(\theta_i) = E_{(s,a,r,s')\sim S(H)}[(r + \gamma \max_{a'} Q(s', a') - Q(s, a))^2],$$

where $S(H)$ is the training history set sampled by the proposed approach, and γ is the discount factor. We use gradient descent, namely Adam algorithm, to optimize the Q-network. In this paper, the discount factor is always set to 1. Because the only reward for our system comes from the predictive performance, so using a fractional discount factor prevents us from selecting more classifier.

The deep Q-network in our model consists of two fully connected hidden layers, where each of them contains 256 nodes. All layers except the output layer are followed by ReLU activation layer. In order to address the overfitting problem, the dropout technique [19] is applied to all but the output layer, and we select 0.5 as the dropout rate.

IV. Results and Analysis

A. Experimental Design

We conduct the experiments on 10 datasets acquired from UCI [20] online data repository. Table I shows the statistics of these datasets.

Our proposed model, instance-based reinforcement learning (IBRL), will be compared against classifiers combination approaches majority voting (MV) and weighted voting (WV), and ensemble selection methods forward selection (FS)[9]. We split the dataset into 10 exclusive groups, 4 of them are used to train the basic classifiers, 4 of them are used to train the RL model, and the rest 2 groups are used for testing. 10 different experiments are conducted on each dataset by switching training and testing groups, the results are averaged with respect to the datasets. We evaluate different models on multiple datasets based on their average accuracies and ranks across all datasets [21].

Dataset	# of instances	# of features	# of labels
abalone	4177	8	29
audiology	226	69	24
breast w	699	9	2
cmc	1473	9	3
credit card	30000	23	2
glass	214	9	6
heart	270	14	2
human activity	10299	561	6
iris	300	4	3
lymphography	148	18	4

TABLE I
DATASETS SUMMARY

We examine the proposed model on both homogeneous and heterogeneous ensembles. In homogeneous cases, we build ensembles that consist of pure decision trees or MLPs. The heterogeneous ensembles are composed of four different types of classifiers: decision trees, MLPs, k-NNs, and naive Bayes. Each type of sub-model contributes roughly $1/4$ of the entire ensemble.

B. Homogeneous Case

The homogeneous ensemble contains only one kind of classifier. We average the results from decision tree ensembles and MLP ensembles. Table II shows the accuracies and ranks for different algorithms selecting ensembles with 100 classifiers.

Although our proposed model obtains the highest accuracy in most of the datasets, the gaps between our approach and both voting mechanisms are relatively small. The fundamental reason is that the classifiers in the ensemble selection models (FS, IBRL) are weaker than those in voting models (MV, WV). Because the ensemble selection models require an additional set of data to train the meta-level algorithm, only a half of data is fed to the base-level models. Another reason that voting mechanisms perform well is the correlation between homogeneous models. Classifiers with the same structure have similar views of data, and consequently have similar predictive behaviours. So, it is unlikely to have neither extraordinary well classifiers nor exceptionally poor ones. As a result, the ensemble selection algorithms will not benefit a lot from removing those underperformed classifiers.

Another pattern shown in the experiment is that the improvement made by our model is more significant in difficult datasets, where the accuracy is less than 80%. This phenomenon can be explained by the binomial distribution. When the average error rate of the classifiers decrease, the probability that more than half of them failed plunges with a much larger extent.

Dataset	Accuracy (Higher is better)				Rank (Lower is better)			
	MV	WV	FS	IBRL	MV	WV	FS	IBRL
abalone	0.2554	0.2536	0.2213	**0.2638**	2.0	3.0	4.0	**1.0**
audiology	**0.7456**	**0.7456**	0.7156	0.7246	**1.5**	**1.5**	4.0	3.0
breast w	0.9607	0.9607	0.9571	**0.9643**	2.5	2.5	4.0	**1.0**
cmc	0.4974	0.5008	0.4534	**0.5330**	3.0	2.0	4.0	**1.0**
credit card	0.8147	0.8145	0.8034	**0.8153**	2.0	3.0	4.0	**1.0**
glass	0.7727	0.7727	0.7575	**0.7878**	2.5	2.5	4.0	**1.0**
heart	0.8271	0.8271	0.7407	**0.8395**	2.5	2.5	4.0	**1.0**
human act	0.9757	0.9757	0.9692	**0.9764**	2.5	2.5	4.0	**1.0**
iris	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**2.5**	**2.5**	**2.5**	**2.5**
lympho	0.8000	0.8000	**0.8222**	0.7555	2.5	**1.0**	2.5	4.0
average	0.7649	0.7651	0.7440	**0.7660**	2.35	2.3	3.7	**1.65**

TABLE II
HOMOGENEOUS ENSEMBLES WITH 100 CLASSIFIERS

Dataset	Accuracy (Higher is better)				Rank (Lower is better)			
	MV	WV	FS	IBRL	MV	WV	FS	IBRL
abalone	0.2736	0.2604	0.2726	**0.2959**	2.0	4.0	3.0	**1.0**
audiology	0.7609	0.7827	0.7827	**0.8696**	4.0	2.5	2.5	**1.0**
breast w	0.9691	0.9707	0.9636	**0.9743**	3.0	2.0	4.0	**1.0**
cmc	0.5795	0.5778	0.5797	**0.5920**	3.0	4.0	2.0	**1.0**
credit card	0.7963	0.7900	**0.8117**	0.8040	3.0	4.0	**1.0**	2.0
glass	0.7273	0.7727	**0.8182**	0.7955	4.0	3.0	**1.0**	2.0
heart	0.8395	0.8889	0.8889	**0.9012**	4.0	2.5	2.5	**1.0**
human act	0.9777	0.9825	0.9864	**0.9874**	4.0	3.0	2.0	**1.0**
iris	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**2.5**	**2.5**	**2.5**	**2.5**
lympho	0.8222	0.8222	0.8667	**0.8889**	3.5	3.5	2.0	**1.0**
average	0.7746	0.7848	0.7971	**0.8109**	3.3	3.1	2.25	**1.35**

TABLE III
HETEROGENEOUS ENSEMBLES WITH 100 CLASSIFIERS

C. Heterogeneous Case

The heterogeneous ensembles contain four different types of classifiers: decision trees, MLPs, k-NNs, and naive Bayes. Table III shows the accuracy and rank for different algorithms on ensembles with 100 classifiers.

Our proposed model out performed the other three approaches in the heterogeneous cases. Unlike homogeneous ensembles, there is little correlation between classifiers with different structures, and the variation within the ensemble is much larger. Because different types of classifiers have different perspective towards the dataset, some structures may have outstanding performance in some particular situation. For example, the k-nearest neighbor classifiers have the highest average accuracy in the glass dataset while the multilayer perceptrons perform the best in the heart dataset. Due to this variation, the range of accuracies of the base-level classifiers is much larger than that in homogeneous ensembles. As a result, combining all sub-models with voting mechanisms does not have advantage to ensemble classifiers: those outstanding classifiers are counteracted by those exceptional poor models. On the contrary, the benefit of eliminating those bad classifier is greater than in homogeneous ensembles. We also perform the two sample t-test [22] on the pairwise ranks of different approaches. We set the null hypothesis as there is no difference between our proposed method and benchmarking methods. With confidence level $p < 0.05$, for all benchmarking approaches, it shows a significant difference which indicates the predominance of our proposed model over other methods.

In most cases, the heterogeneous ensemble methods have higher accuracy than the homogeneous ensemble methods. The average accuracy of heterogeneous ensembles is 1.5% higher than the homogeneous ensembles. The ensemble selection algorithms usually focus on those base-level models that have high accuracy. Therefore, we compare the top one quarter of all classifiers among homogeneous ensembles and heterogeneous ensembles. The difference is larger that heterogeneous ensembles obtain a 2.6% advantage over the homogeneous ensembles. An intuitive interpretation is that homogeneous models have similar structure, so they tend to make similar mistakes. Consequently, if an instance cannot be correctly predicted by one classifier, it may not be properly classified by all others. Heterogeneous models have different performance on a particular instance, so if a classifier failed to predict on an instance, some others may be able to complement.

Dataset	Accuracy (Higher is better)				Rank (Lower is better)			
	MV	WV	FS	IBRL	MV	WV	FS	IBRL
abalone	**0.2535**	0.2448	0.2289	0.2488	**1.0**	3.0	4.0	2.0
audiology	0.7934	0.7934	0.7717	**0.8043**	2.5	2.5	4.0	**1.0**
breast w	0.9476	0.9571	0.9524	**0.9619**	4.0	2.0	3.0	**1.0**
cmc	0.4416	0.4687	0.4653	**0.4974**	4.0	2.0	3.0	**1.0**
credit card	0.7825	0.7853	0.7968	**0.7934**	4.0	3.0	2.0	**1.0**
glass	0.6954	0.7065	**0.7113**	0.6975	4.0	2.0	**1.0**	3.0
heart	0.7901	**0.8148**	0.8025	0.8025	4.0	**1.0**	2.5	2.5
human act	0.9264	0.9229	**0.9334**	0.9302	3.0	4.0	**1.0**	2.0
iris	0.9433	0.9433	0.9632	**0.9646**	4.0	3.0	2.0	**1.0**
lympho	0.7111	0.6888	0.7111	**0.8222**	2.5	4.0	2.5	**1.0**
average	0.7285	0.7326	0.7337	**0.7523**	3.3	2.65	2.5	**1.55**

TABLE IV
PARTIAL DATA WITH 50 CLASSIFIERS

D. Partial Dataset

Ensemble method is a widely applied solution when models cannot be efficiently trained on the entire dataset. Many real world tasks may be involved in these situations:

- Dataset is geographically divided due to economic or security issues [23].
- Dataset is temporally distributed such as streaming data [24], [25].
- Some datasets are just too big to fit into one single model [26], [27].

In order to simulate this situation, we split training data into three different parts, where each part contains all data instances but with only one third of the features randomly, and different parts share no common feature. Each base-level classifier is then trained using a single portion of data. Table [8] shows the average accuracy of different algorithms on ten datasets. These experiments are conducted on both homogeneous and heterogeneous ensembles with 50 classifiers.

Comparing to the heterogeneous ensembles, all algorithms suffer from universal decrease in accuracy. However, our proposed model is the most robust one that endures the least amount of decrease. The average rank of our model is better in this experiment than the heterogeneous experiments. We perform the two sample t-test on the pairwise average ranks of different algorithms. Our proposed model is compared against each of the benchmarking technique individually with the null hypothesis that those algorithms are identical. With confidence level $p < 0.05$, the tests show significant difference between our proposed model and voting techniques (MV, WV). With confidence level $p < 0.1$, the tests show critical difference between our model and FS. Therefore, the IBRL model is an ideal alternative in the distributed environment, where accuracy can be somewhat sacrificed in order to reduce the amount of data transportation.

V. CONCLUSION

Our proposed model, instance-based reinforcement learning, tackles the ensemble selection problem. It essentially solves a searching problem in large state spaces using reinforcement learning with deep Q-network. Our proposed model is the first ensemble selection method that provides instance-based selection. The experimental results indicate that the instance-based characteristic ensures stronger performances in predicting tasks compared to traditional ensemble selection methods that consult fixed set of classifiers. We also show that ensemble selection method, especially ours, have outstanding performances in selecting heterogeneous ensembles, which is a more preferable form than the homogeneous ensembles. Our model is scalable, and it is more adaptable to distributed databases. Due to the environment free characteristics of reinforcement learning, our model is capable to solve various real world problems with some minute changes.

We also study on the partitioned datasets, which is a simulation of the distributed data storage. The experimental results show that our IBRL model is the most robust one that suffered from acceptable reduction in accuracy. This performance prove that our proposed model can be scaled up easily in large and distributed environment. Although we define the rewards solely based on the final result, the definition of costs and rewards are extremely flexible. This feature enables our model to be adaptable to many real world scenarios. For example, this model can be applied to the specialist consulting problem where each specialist is regarded as a sub-model and the consultation comes with a certain price.

One drawback of our model is the sequential computation for ensemble selection in validation stage. Because of that, our model has to sacrifice some efficiency for accuracy, which is a classic trade-off in machine learning field. A potential solution to this problem is the batch consulting to classifiers. That is, instead of visiting a single classifier at once, we choose a bunch of sub-models and read the prediction from them. This modification introduces a different transition form to the Markov Decision Process. Moreover, this approach leads to a much larger action space, a larger neural network, and consequently, a longer training time.

REFERENCES

[1] T. G. Dietterich, "Machine-learning research," *AI magazine*, vol. 18, no. 4, pp. 97–97, 1997.

[2] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 10, pp. 993–1001, 1990.

[3] S. Džeroski and B. Ženko, "Is combining classifiers with stacking better than selecting the best one?" *Machine learning*, vol. 54, no. 3, pp. 255–273, 2004.

[4] Y. Freund, "An adaptive version of the boost by majority algorithm," *Machine learning*, vol. 43, no. 3, pp. 293–318, 2001.

[5] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[6] I. Partalas, G. Tsoumakas, and I. Vlahavas, "Pruning an ensemble of classifiers via reinforcement learning," *Neurocomputing*, vol. 72, no. 7-9, pp. 1900–1909, 2009.

[7] D. H. Wolpert, "Stacked generalization," *Neural networks*, vol. 5, no. 2, pp. 241–259, 1992.

[8] C. Tamon and J. Xiang, "On the boosting pruning problem," in *European conference on machine learning*. Springer, 2000, pp. 404–412.

[9] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes, "Ensemble selection from libraries of models," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 18.

[10] D. D. Margineantu and T. G. Dietterich, "Pruning adaptive boosting," in *ICML*, vol. 97. Citeseer, 1997, pp. 211–218.

[11] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[12] B. Efron, "Bootstrap methods: another look at the jackknife," in *Breakthroughs in statistics*. Springer, 1992, pp. 569–593.

[13] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[14] A. M. Canuto, L. M. Oliveira, J. C. Xavier, A. M. Santos, and M. C. Abreu, "Performance and diversity evaluation in hybrid and non-hybrid structures of ensembles," in *Fifth International Conference on Hybrid Intelligent Systems (HIS'05)*. IEEE, 2005, pp. 6–pp.

[15] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[16] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[17] C. Dimitrakakis and S. Bengio, "Online adaptive policies for ensemble classifiers," *Neurocomputing*, vol. 64, pp. 211–221, 2005.

[18] M. G. Lagoudakis and M. L. Littman, "Algorithm selection using reinforcement learning." in *ICML*. Citeseer, 2000, pp. 511–518.

[19] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[20] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[21] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine learning research*, vol. 7, no. Jan, pp. 1–30, 2006.

[22] Student, "The probable error of a mean," *Biometrika*, pp. 1–25, 1908.

[23] A. Prodromidis, P. Chan, S. Stolfo *et al.*, "Meta-learning in distributed data mining systems: Issues and approaches," *Advances in distributed and parallel knowledge discovery*, vol. 3, pp. 81–114, 2000.

[24] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavaldà, "New ensemble methods for evolving data streams," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 139–148.

[25] W. N. Street and Y. Kim, "A streaming ensemble algorithm (sea) for large-scale classification," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2001, pp. 377–382.

[26] L. Hall, K. Bowyer, W. Kegelmeyer, T. Moore, and C.-m. Chao, "Distributed learning on very large data sets," in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Citeseer, 2000, pp. 79–84.

[27] Y. Zhang and S. Bhattacharyya, "Genetic programming in classifying large-scale data: an ensemble method," *Information Sciences*, vol. 163, no. 1-3, pp. 85–101, 2004.