

LEARNING PARSIMONIOUS ENSEMBLES FOR UNBALANCED COMPUTATIONAL GENOMICS PROBLEMS

ANA STANESCU and GAURAV PANDEY*

*Icahn Institute for Genomics and Multiscale Biology and
Department of Genetics and Genomic Sciences, Icahn School of Medicine at Mount Sinai
New York, NY, USA*

*E-mail: gaurav.pandey@mssm.edu

Prediction problems in biomedical sciences are generally quite difficult, partially due to incomplete knowledge of how the phenomenon of interest is influenced by the variables and measurements used for prediction, as well as a lack of consensus regarding the ideal predictor(s) for specific problems. In these situations, a powerful approach to improving prediction performance is to construct ensembles that combine the outputs of many individual base predictors, which have been successful for many biomedical prediction tasks. Moreover, selecting a *parsimonious* ensemble can be of even greater value for biomedical sciences, where it is not only important to learn an accurate predictor, but also to interpret what novel knowledge it can provide about the target problem. Ensemble selection is a promising approach for this task because of its ability to select a collectively predictive subset, often a relatively small one, of all input base predictors. One of the most well-known algorithms for ensemble selection, CES (Caruana *et al.*'s Ensemble Selection), generally performs well in practice, but faces several challenges due to the difficulty of choosing the right values of its various parameters. Since the choices made for these parameters are usually ad-hoc, good performance of CES is difficult to guarantee for a variety of problems or datasets. To address these challenges with CES and other such algorithms, we propose a novel heterogeneous ensemble selection approach based on the paradigm of reinforcement learning (RL), which offers a more systematic and mathematically sound methodology for exploring the many possible combinations of base predictors that can be selected into an ensemble. We develop three RL-based strategies for constructing ensembles and analyze their results on two unbalanced computational genomics problems, namely the prediction of protein function and splice sites in eukaryotic genomes. We show that the resultant ensembles are indeed substantially more parsimonious as compared to the full set of base predictors, yet still offer almost the same classification power, especially for larger datasets. The RL ensembles also yield a better combination of parsimony and predictive performance as compared to CES.

Keywords: Heterogeneous ensembles; Ensemble selection; Reinforcement learning; Computational genomics.

1. Introduction

Prediction problems in biomedical sciences, such as protein function prediction,^{1,2} drug target discovery,³ and classification of genomic elements⁴ are generally quite difficult. This is due in part to incomplete knowledge of how the phenomenon of interest is influenced by the variables and measurements used for prediction, as well as a lack of consensus regarding the ideal predictor(s) for specific problems. From a data perspective, the frequent presence of extreme class imbalance, missing values, heterogeneous data sources of different scales, overlapping feature distributions, and measurement noise further complicate prediction.

In scenarios like these, a powerful approach to improving prediction performance is to construct ensemble predictors that combine the output of many individual base predictors.^{5,6} These ensembles have been very successful in producing accurate predictions for many biomedical prediction tasks.⁷⁻¹³ The success of these methods is attributed to their ability to reinforce accurate predictions as well as correct errors across many diverse base predictors.¹⁴ Diversity among the base predictors is key to ensemble performance: If there is complete consensus (no diversity), the ensemble does not provide any advantage over any of the base predictors. Similarly, an ensemble lacking any consensus (highest diversity) is unlikely to produce confident predictions. Successful ensemble methods strike a balance between diversity and accuracy.^{15,16} Popular methods like boosting¹⁷ and random forest¹⁸ generate this diversity by sampling from or assigning weights to training examples. However, they generally utilize a single type of base predictor, such as decision trees, to build the ensemble. Such *homogeneous* ensembles may not be the best choice for problems in biomedical sciences, where the ideal base prediction method is often unclear due to incomplete knowledge and/or data issues.

A more potent approach in this scenario is to build ensembles from the predictions of a wide variety of *heterogeneous* base predictors. Two commonly used heterogeneous ensemble methods are *stacking*,^{19,20} and

ensemble selection.^{21,22} Recently, we showed that these methods are more effective than homogeneous ensembles and individual classification methods for complex prediction problems in genomics.^{23,24} Other studies have produced similar results.^{8,25}

Ensemble selection is an especially promising approach, not only for improving prediction performance, but also because of its ability to select a collectively predictive subset, often a relatively small one, of all input base predictors. This ability to select a *parsimonious* ensemble can be very valuable for biomedical sciences, where it is not only important to learn an accurate predictor, but also to interpret what novel knowledge it can provide about the target problem. For instance, in predicting protein function,¹ it is critical to identify the biological features or principles on the basis of which accurate predictions of protein function are made.² It would be easier to reverse engineer a smaller (more parsimonious) ensemble to identify such features or principles than a much larger one, such as all the base predictors taken together. This goal motivated us to develop better algorithms for ensemble selection.

The most well-known algorithm for ensemble selection, which we will refer to as CES (Caruana *et al.*'s Ensemble Selection),^{21,22} iteratively grows an ensemble by adding base predictors that produce a gain in prediction performance by (indirectly) enhancing the diversity of the ensemble. Although CES generally performs well in practice, it faces several challenges due to the difficulty of choosing the right values of its various parameters (for details, refer to Section 2.2). For instance, it is unclear how many base predictors should comprise the final ensemble of CES, how many (one or more) should be added in each iteration of the algorithm, and what the right termination condition should be. Since the choices made for these parameters are usually ad-hoc, good performance of CES is difficult to guarantee for a variety of problems or datasets.

To address these challenges with CES and other such algorithms, we propose a novel heterogeneous ensemble selection approach based on the well-established paradigm of reinforcement learning (RL).²⁶ We demonstrate how RL offers a more systematic and mathematically sound methodology for exploring the many possible combinations of base predictors that can be selected into an ensemble. We test our proposed approaches, and several baselines, on two important computational genomics problem, namely the prediction of protein function and splice sites in eukaryotic genomes. We focus on these unbalanced problems as effective individual (base) predictive models are difficult to learn for them, thus offering a suitable use case for testing ensemble predictors. Our results show that the RL ensembles are indeed substantially more parsimonious with respect to the full set of base predictors, but still offer almost the same predictive power, especially for larger datasets. The RL ensembles also yield a better combination of parsimony and predictive performance as compared to CES. We expect our approaches to be effective for other biomedical problems as well as aid in interpretability, although the latter is a challenging and often subjective task for complex problems. Thus, interpretation of the ensembles we discover is outside the scope of this work.

2. Preliminary Materials and Methods

2.1. Problem definitions and datasets

We assess the predictive ability of various ensemble (selection) techniques, such as CES and our RL-based ones, on several protein function (PF) and splice site (SS) prediction datasets.

Protein Function Prediction: Gene expression data are commonly used for predicting protein function, as the simultaneous measurement of gene expression across the entire genome enables effective inference of functional relationships and annotations.^{1,2} Thus, for the PFP assessment, we use the gene expression compendium of Hughes *et al.*²⁷ to predict the functions of roughly 4,000 *S. cerevisiae* genes. Among these genes, the three most abundant functional labels (GO terms) from the list of most biologically interesting and actionable Gene Ontology Biological Process terms compiled by Myers *et al.*²⁸ are used in our evaluation. These labels are GO:0051252 (regulation of RNA metabolic process), GO:0006366 (transcription from RNA polymerase II promoter) and GO:0016192 (vesicle-mediated transport). We refer to these prediction problems as PF1, PF2, and PF3 respectively (details in Table 1).

Prediction of Splicing Sites: RNA splicing is a naturally occurring phenomenon that contributes to protein diversity. Generally, when creating mature RNA from DNA, introns are removed (or spliced out) from the

Table 1. Details of protein function (PF) and splice site (SS) datasets, including the number of features, number of examples in the minority (positive) and majority (negative) classes, and total number of examples.

| Problem | Protein Function Datasets (PF) | | | Splice Site Datasets (SS) | | | | |
|------------|--------------------------------|---------------------------------|-------|---------------------------|-------------------|---------------------|-------------------|--------------------|
| | PF1 | PF2 (<i>S. cerevisiae</i>) | PF3 | <i>D. melanogaster</i> | <i>C. elegans</i> | <i>P. pacificus</i> | <i>C. remanei</i> | <i>A. thaliana</i> |
| #Features | 300 | 300 | 300 | 141 | 141 | 141 | 141 | 141 |
| #Positives | 382 | 344 | 327 | 1,598 | 997 | 1,596 | 1,600 | 1,600 |
| #Negatives | 3,597 | 3,635 | 3,652 | 158,150 | 99,003 | 156,326 | 157,542 | 158,377 |
| #Total | 3,979 | 3,979 | 3,979 | 159,748 | 100,000 | 157,922 | 159,142 | 159,977 |

gene sequence and exons are retained (or transcribed). Splice sites are conserved nucleotide dimers found at the interfaces between exons and introns. In general, splice sites are *canonical*, as acceptor splice sites are signaled by the occurrence of the consensus dimer “AG” at the 3’ end of the intron, while donor splice sites are characterized by the consensus dimer “GT”, situated at the 5’ end of the intron. Such dimers occur frequently throughout most eukaryotic genomes but their presence alone is not sufficient to declare a splice site. Correctly identifying splice sites is an essential step towards genome annotation, and a difficult problem due to the highly unbalanced ratio of bona fide splice sites to decoy dimers.²⁹ In this work, we focus on identifying acceptor splice sites. In machine learning terms, the problem is formulated as a binary classification of DNA sequences (141-nucleotide-long windows around “AG” dimers, with the dimer situated at position 61) as true acceptor splice sites and decoy dimers. Thus, we assess the ability of ensemble learning to address this important problem on five datasets of acceptor splice sites from five organisms: *D. melanogaster*, *C. elegans*, *P. pacificus*, *C. remanei*, and *A. thaliana*, published by Schweikert *et al.*⁴ and Ratsch *et al.*³⁰

Note that in some of our experimental evaluations, we investigate the PF and SS datasets results separately due to the substantially different numbers of examples in these datasets.

2.2. Ensemble selection with CES

Caruana *et al.*’s Ensemble Selection (CES)^{21,22} is arguably the most well-known ensemble selection method. CES begins with an ensemble consisting of the best n individual base predictors ($n = 1$ in our implementation), and iteratively adds new predictors that maximize the performance of the resultant ensemble on a validation set according to a chosen measure. In each iteration, a pool of m of the candidate base predictors are selected randomly without replacement ($m = \#base\ predictors$ in our implementation²²), and the performance resulting from the addition of each individual candidate to the current ensemble is evaluated. The candidate resulting in the best ensemble performance is selected, the ensemble is updated with it, and the process repeats until a maximum ensemble size (set to the *total #base predictors*²² in our implementation) is reached. We also varied the values of the pool size (m) and maximum size to test the sensitivity of CES to these parameters.

2.3. Reinforcement Learning (RL)

The RL machine learning paradigm²⁶ allows decision-making software agents to learn the ideal behavior within a specific observable environment such that their performance at the specified task is maximized. In order to learn its behavior, an agent requires feedback for its actions, given by the environment in the form of reinforcement signals. Learning what the best action an agent can take given the current state is achieved through trial-and-error interactions with the environment. One of the most basic applications of RL is teaching a novice robot how to traverse a room from one end to another with various obstacles being placed at random locations in the room; for more complex robotics tasks refer to Kober *et al.*³¹

Generally, RL problems are formulated in terms of Markov Decision Processes (MDPs),³² a commonly adopted framework for modeling environments and sequential decision making. The environment is made up of a finite set of states S , and the actions come from a discrete set A of actions allowed in a given state. The agent’s job is to investigate the environment by taking actions and observing the rewards. The Markov property affirms that the current state contains enough information to make a decision about the next action. The goal of RL is to repeat the action-observation process that results in the agent learning a good/optimal strategy, called “policy”, for collecting rewards, and completing the task(s) at hand.

Since there is no prior knowledge about any rewards, nor any transition probabilities from any states, (in other words, there is no model available), the agent has to actively “sample” the MDP. Hence the need for *exploration*: the agent tries different actions, often previously untried ones, and assesses their outcome. However, in order to gain sufficient cumulative reward, the agent must also *exploit* its current knowledge about actions already tested that have proved to be beneficial. This exploration/exploitation balance is critical, yet difficult to determine, and represents a fundamental dilemma in RL scenarios. This balance is usually enforced by the classic ϵ -greedy strategy:³³ with probability ϵ , the agent takes a randomly selected action (exploration), and with probability $1 - \epsilon$, the agent chooses the action with the highest estimated payoff (exploitation).

2.4. Q-Learning

Following the ϵ -greedy exploration mechanism described above, the agent is able to gather enough information about the environment and create its own model, more specifically, to estimate the quality of each state-action combinations; this mapping is known as the Q-value function. The agent takes an action a_t in a state s_t , ends up in state s_{t+1} , and receives a reward r_t ; subsequently, the Q-value associated with action a_t in state s_t is updated. One of the most popular ways for estimating Q-value functions in a model-free framework is the Q-learning algorithm.³⁴ Under specific conditions/assumptions, Q-learning is able to find an optimal policy (π) regardless of the model the agent adopts, *i.e.*, which action a_t it takes in state s_t , provided it tries all actions of all states infinitely many times. The policy, which is the agent’s learned way of behaving in the environment, is estimated by continuously updating the action-value function $Q(s_t, a_t)$, as described in Equation 1. Here, the learning rate α controls how quickly the learning occurs, and the discount factor γ controls how important future rewards are.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \cdot \left(r_{t+1} + \gamma \cdot \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right) \quad (1)$$

3. Proposed Approach

Although CES (Section 2.2) represents an intuitive and straight-forward approach to ensemble selection, several of its inputs/parameters are very difficult to specify beforehand, which is very likely to impact its performance adversely. For instance, the best values of parameters m and n are difficult to specify a priori for a given dataset/problem. Similarly, the termination criteria, *i.e.*, when the CES ensemble should stop growing so as to avoid overfitting, are also often unclear, and are often set in an ad-hoc manner. Due to these factors, CES is only able to perform an ad-hoc sub-optimal search of the possible ensemble space, and not an optimized exhaustive one. We address these challenges of CES and make ensemble selection more rigorous and exhaustive by leveraging concepts from RL (Section 2.3). In particular, we take advantage of the Q-learning algorithm (Section 2.4), which is proven to converge to the optimal solution/policy.^{34,35}

To formulate the ensemble selection problem as an RL task, it is necessary to define the following key components of the model. The agent is our proposed ensemble selection algorithm. We define the environment as a deterministic one, in the sense that taking the same action in the same state on two different occasions cannot result in different next states and/or different reinforcement values. More specifically, the environment in which our agent operates consists of all possible subsets of the n base predictors, each serving as a possible ensemble, thus consisting of 2^n states. The environment includes the empty set, which is considered the start state in our implementation. An example environment generated by five base predictors is shown in Fig. 1. It can be viewed as a lattice, and the arrows represent the actions the agent is allowed to take at each state.

The agent investigates the environment by moving from one state (one ensemble) to another in search of better rewards. The reward $R(s_t, a_t, s_{t+1})$ received for the transition from the state s_t to the new state s_{t+1} by executing the action a_t is calculated based on the predictive performance of the ensemble(s) involved in the action. In our experiments, performance is assessed on a validation set separate from the training set, as explained in Section 4. The agent begins its learning at the start state (which corresponds to the empty set) without any prior knowledge about the subsequent states or any of the rewards. Then, the agent moves downward through the lattice from one state to another, until it reaches the final state, *i.e.*, the full ensemble

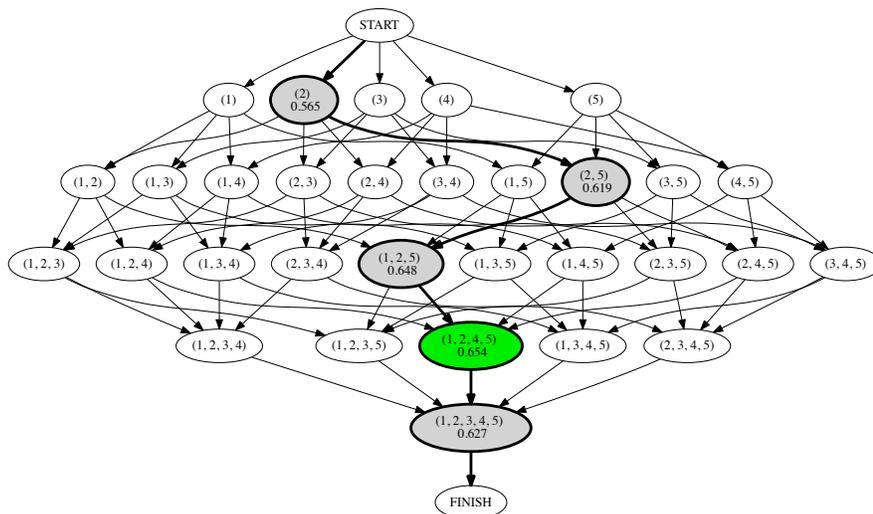


Fig. 1. Example of an environment constructed from five base predictors for the *P. pacificus* splice site dataset. The numbers in the nodes represent the predictive performance of the corresponding ensemble in terms of F-max on the validation set. The path highlighted in grey represents the policy found at the end of the learning process using the RL_greedy strategy ($\epsilon = 0.01$ and ten consecutive episodes yield the same ensemble). The state with the highest reward (*i.e.*, the highest predictive performance) along the path is (1, 2, 4, 5) (green), which is returned by the RL_greedy as the resulting ensemble for the illustrated example. The nodes (2, 4, 5) and (1, 2) represent the ensembles returned by the RL_pessimistic and RL_backtrack strategies respectively. Figure created with Graphviz.³⁶

containing all base predictors. Each action is equivalent to adding exactly one other base predictor to the ensemble. In our model, the agent cannot “jump” from a state of k base predictors to a state of $k + 1$ base predictors that is not directly connected in the lattice, such as from (2, 3) to (1, 3, 4). This is necessary to control the complexity of the state-action space. A “traversal” of the lattice from the start state to the finish state is referred to as a “learning episode”. We propose Q-learning-based algorithms that employ three different search strategies formulated as different ways of constructing learning episodes and computing rewards. These strategies essentially represent various models of the environment and the interactions between the agent and the environment. The goal, *i.e.*, traversing the lattice and ultimately reaching the full ensemble state, remains the same. These strategies are explained below.

3.1. Greedy strategy (RL_greedy)

Our first strategy emulates a “greedy” agent, whose goal is to reach the full ensemble quickly. As a consequence of this plan, the agent rapidly accumulates new base predictors and greedily constructs the ensemble. With each step taken from state s_t to s_{t+1} in the environment, the agent receives state s_{t+1} ’s performance as a reward, *i.e.*, $R(s_t, a_t, s_{t+1}) = f(s_{t+1})$, and updates its Q-table as per Equation 1. Each learning episode has a fixed number of steps determined by the depth of the lattice. Fig. 1 shows an example of the path, as well as the resultant ensemble, that RL_greedy finds from a sample lattice constructed from five base predictors.

3.2. Pessimistic strategy (RL_pessimistic)

The second strategy resembles a “pessimistic” agent that resets itself to the start state as soon as a less than desirable state is visited, without finishing the traversal of the lattice, and starts a new learning episode. Thus, the lengths of the learning episodes may vary, depending on how soon the agent encounters a “depreciated” state. This strategy is based on the hypothesis that such a depreciated state (negative reward) might indicate a path of overfit and/or under-performing ensembles. For this strategy, the reward is calculated as the difference in performance from s_t to s_{t+1} , *i.e.*, $R(s_t, a_t, s_{t+1}) = f(s_{t+1}) - f(s_t)$.

3.3. Backtrack strategy (*RL_backtrack*)

The last strategy, *RL_backtrack*, uses the same reward function as *RL_greedy*. However, unlike the latter strategy, which aims to reach the full ensemble state quickly, the agent “backtracks” (goes back) to the immediately previous state as soon as a decrease in performance is encountered, and resumes its learning from there. A feature of this strategy is that the learning episodes can have a variable numbers of steps, as the agent might wander inside the lattice until it finds an acceptable path ending in the full ensemble.

For all the strategies, the policy derived from the learned action-value function yields a sequence/path of increasingly larger ensembles. We choose the ensemble on this path with the highest performance as the selected ensemble to be evaluated on the test set.

4. Experimental Setup

In all our RL-based experiments, the parameters of the Q-learning algorithm described in Section 2.4, namely α and γ are set to 0.1 and 0.9 respectively, which are commonly used values.²⁶ The exploration/exploitation trade-off is controlled by the ϵ probability discussed in Section 2.3. A higher probability indicates more exploration. In our work, we experiment with $\epsilon \in \{0.01, 0.1, 0.25, 0.5\}$ for the PF datasets and with $\epsilon \in \{0.01, 0.1, 0.25\}$ for the larger SS ones. The iterative nature of Q-learning requires the initialization of its parameters (values in the Q-table). We initialize the Q-table as a *zero* matrix, and update the values as states and rewards are observed by the agent, as guided by the search strategies defined above.

Although convergence of the Q-learning algorithm and the final policy “optimality” are theoretically guaranteed,^{34,35} and achieved when the agent visits all of the environment’s states infinitely often, we adopt more practical termination criteria for our search strategies in our experiments. For *RL_greedy*, the stopping point is reached when the policy induced by the agent produces the same result (*i.e.*, the same ensemble with the highest performance within the currently selected policy) for ten consecutive episodes. In contrast, *RL_pessimistic* and *RL_backtrack* are susceptible to longer running times, and even oscillations within the lattice and subsequently of the Q-values learned. For this reason, we set the termination criterion for these strategies as when the agent has taken a fixed number of steps in the environment, specifically 0.5 million. Note that these practical assumptions make it difficult for us to assess the theoretical optimality of our RL algorithms and their results.

In order to assess the relative performance of our RL-based ensemble selection strategies, we also employ several baselines. First, we consider the best base predictor (BP) of each initial set of base classifiers, which is the classifier with the highest classification prediction performance on the validation set. At the opposite end of the spectrum, the full ensemble (FE), consisting of all initial base classifiers, will be the largest ensemble, and our second baseline. Finally, the third baseline is the ensemble produced by CES, implemented as described in Section 2.2.

The general workflow used for the experimental evaluation of the above approaches is shown in Fig. 2. We use 5-fold cross validation to estimate the performance of all the models. All base predictors are learned on the training set (60% of the original data described in Table 1), which is balanced using undersampling of the majority class. The validation set (20% of the data) is used for calculating the rewards of the nodes in the RL environment, and to estimate the performance of the candidate base predictors in the CES approach. The test set (comprising the remaining 20% of the data) is used to assess the overall performance of all studied algorithms. An experiment for an algorithm being tested consists of the collection of these performance scores over all five rounds of this cross-validation.

All performance evaluation, whether internal (on the validation set) or external (on the test set) is conducted using F-max, which is the maximum value of the F-measure across all the values of precision and recall at many thresholds applied to the prediction scores generated by the base classifiers and the resultant ensembles. F-max is appropriate given the highly skewed class distributions of the datasets used in our study, and has been shown to be reliable for performance evaluation in a recent large-scale assessment of protein function prediction.² Other metrics that are sensitive to unbalanced problems, *e.g.*, area under the Precision-Recall curve, can also be used.

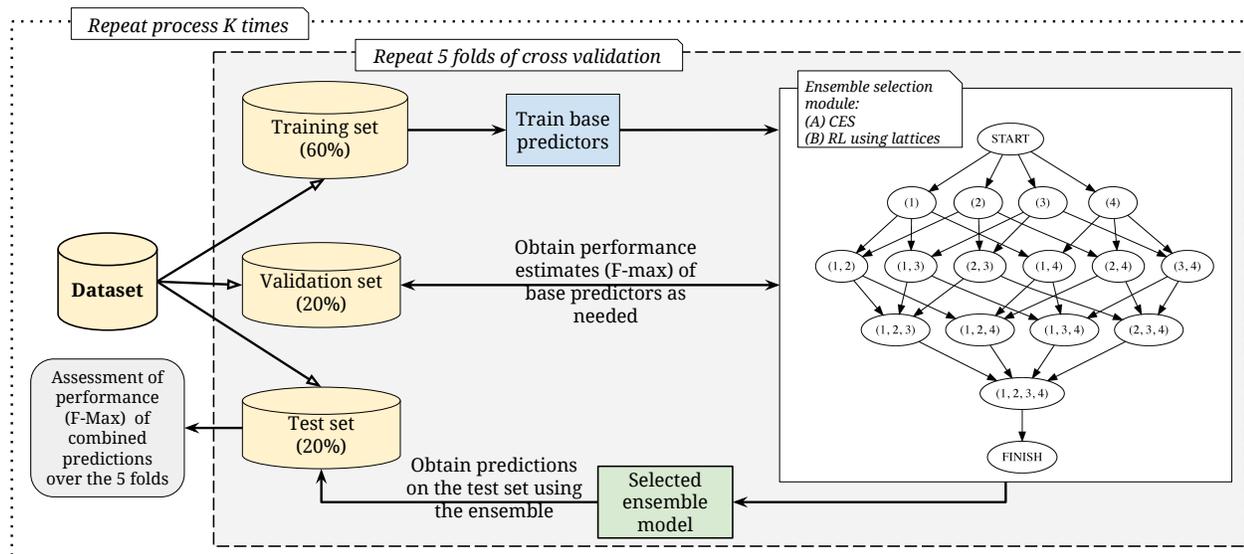


Fig. 2. Visual description of the workflow used to in our experiments. $K = 10$ for the PF datasets and $K = 5$ for the SS datasets.

The ensembles selected by the various approaches we tested (BP, RL, CES) are created by combining the probabilities produced by the constituent base predictors using a weighted average. Here, the importance (weight) of each base predictor is proportional to its predictive performance (measured in terms of the F-max score) on the validation set for all the approaches. We also considered options such as unweighted mean and median but observed that the performance of the ensembles was suffering because of the worst performing individual base predictors among them. In order to efficiently obtain the reward of each state or the performance of the ensemble being considered, we aggregate the predictions using a cumulative moving average.

We train 18 diverse base classification algorithms from Weka,³⁷ including Naïve Bayes, Multilayer Perceptron, SVM with a polynomial kernel, AdaBoost, Logistic Regression, and Random Forest. Each training set is resampled – to balance the classes – with replacement 10 times, resulting in 180 base predictors. Each ensemble selection algorithm is presented with a pool of base predictors (classification models). We start all our experiments with ten base predictors and increase this set gradually, with steps of ten randomly selected base predictors for each experiment, until we reach the entire set of 180 base predictors. This setup is designed to address the question of how the ensemble selection methods behave with an increasingly larger set of initial base predictors to select from. The performance of the ensembles resulting from all these methods was evaluated across all these sizes, resulting in curves depicting the dependence of F-max on the number of initial base classifiers. To account for variation, each set of experiments was repeated ten times for the protein function datasets and five times for the splice site datasets (due to time constraints and the much larger size of the SS datasets). We used the area under these curves, denoted auESC (area under Ensemble Selection Curve), as a global evaluation measure for the various algorithms in our study, as it provides a global assessment of ensemble performance across a variety of base predictors. The area is normalized by its maximum possible value, which is the total number of base predictors (the maximum possible value of F-max is one); thus, the maximum value of auESC is one. However, this metric does not follow the same characteristics as auROC, such as random predictors/ensembles producing a score of 0.5. Therefore, auESC is mostly intended for comparative analyses between algorithms running on the same datasets (as done in our experiments), rather than for assessing the absolute performance of these algorithms.

5. Results

In this section, we will investigate various dimensions of classification performance and parsimony of ensembles constructed for the protein function (PF) and splice site (SS) prediction problems.

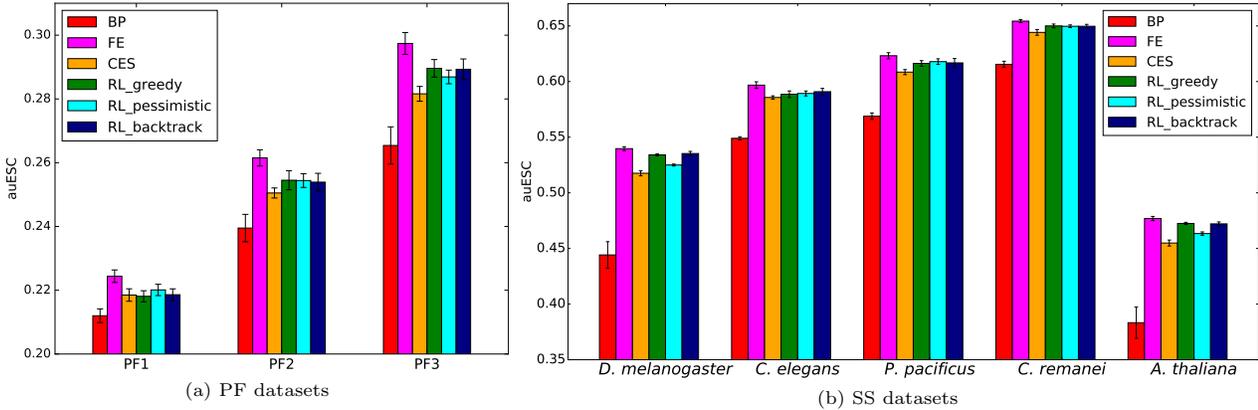


Fig. 3. Overall performance (as the auESC score described in Section 4) of all tested algorithms, evaluated across 18 sizes of initial base predictors (from ten to 180, in steps of ten), for (a) PF and (b) SS datasets. The standard errors are calculated over ten repetitions of each experiment for the PF datasets and five repetitions for the SS datasets. The first three bars of each dataset belong to the baselines considered, namely BP (best single base predictor), FE (full ensemble), and the CES algorithm. The last three bars of each dataset represent the RL-based approaches, namely RL-greedy (ten consecutive episodes yielding the same ensemble), RL-pessimistic, and RL-backtrack (both with 0.5 million training steps). For all RL-based approaches, the exploration/exploitation probability $\epsilon = 0.01$.

5.1. Overall performance of ensemble selection methods

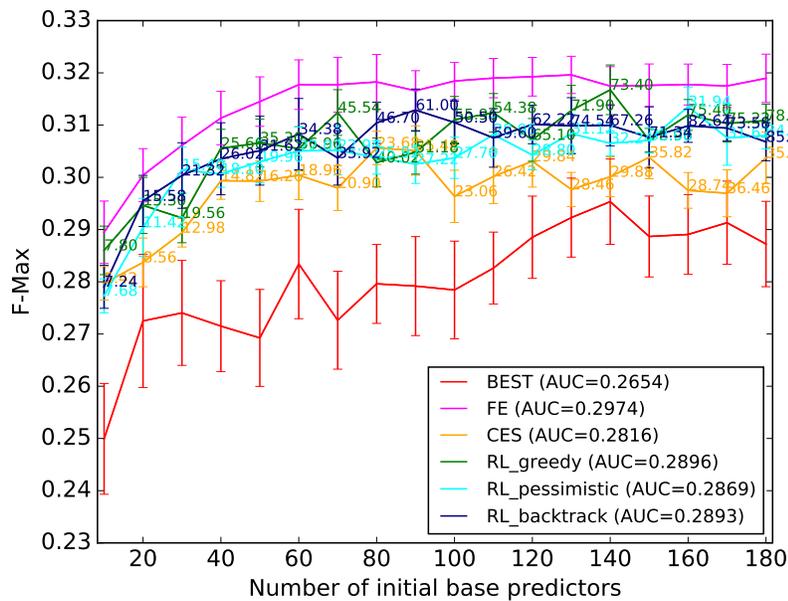
We first compared the overall classification performance of all the tested ensemble selection algorithms on all the datasets considered. These results are presented in the form of auESC scores in Fig. 3. We also tested the statistical significance of the comparisons of these algorithms in terms of these scores using the Friedman-Nemenyi tests.³⁸ Several trends can be observed from these figures.

- The best single base predictor (BP) is consistently outperformed by the full ensemble (FE $p = 6.805 \times 10^{-20}$ for the PF datasets and $p = 2.59 \times 10^{-15}$ for the SS datasets) and the RL-based approaches ($p_{RL_greedy} = 3.29 \times 10^{-4}$, $p_{RL_pessimistic} = 1.33 \times 10^{-6}$, $p_{RL_backtrack} = 7.87 \times 10^{-5}$ for the PF datasets and $p_{RL_greedy} = 5.27 \times 10^{-9}$, $p_{RL_pessimistic} = 1.105 \times 10^{-5}$, $p_{RL_backtrack} = 2.78 \times 10^{-8}$ for the SS datasets), thus validating the benefit of ensembles for these problems. In contrast, CES does not show statistically significant improvement over BP ($p > 0.05$ for both types of datasets).
- The biggest ensemble consisting of all the base predictors together (FE) achieves the highest performance across all tested datasets ($p < 0.05$ for pairwise comparisons with all other approaches.)
- For the smaller PF datasets, CES and the RL-based approaches are comparable ($p > 0.05$ for all pairwise comparisons). For the much larger SS datasets, the RL-based RL-greedy and RL-backtrack approaches perform significantly better than CES ($p_{RL_greedy} = 0.01$, $p_{RL_backtrack} = 0.025$), while RL-pessimistic does not ($p > 0.05$).
- Among the RL-based approaches, RL-greedy produces the best overall performance in terms of auESC, but the performance of all these approaches is statistically comparable ($p > 0.05$ for all pairwise comparisons for both PF and SS datasets.)

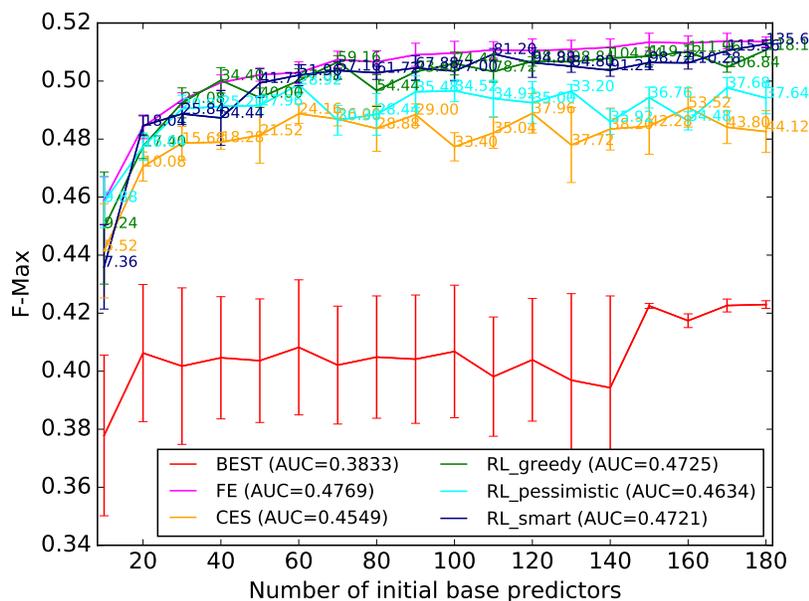
5.2. Detailed examination of ensemble characteristics over various ensemble sizes

The analysis based on auESC values shows the overall comparison of the classification performance of the approaches. However, it is critical to examine in detail the dynamics of these ensembles as the number of initial base predictors increases, and as a result, the sizes of the ensembles learnt. Due to lack of space, we only show two representative sets of curves in Fig. 4, one for the PF3 dataset and another for the splice site dataset from *A. thaliana*, showing the variation of ensemble F-max as the number of base predictors increases. These datasets were selected as representatives, as the ensembles showed the most benefit for them over individual base predictors (Fig. 3).

Fig. 4 shows that the performance of several ensemble approaches improves as the number of initial base



(a) PF3



(b) Splice site dataset from *A. thaliana*

Fig. 4. Performance in terms of F-max of all ensembles as the number of initial base predictors increases from ten to 180 in steps of ten. Each curve corresponds to averages over ten repetitions of each experiment for the (a) PF3 dataset and five repetitions for the (b) *A. thaliana* splice site dataset, along with standard error bars. The curves are annotated with average sizes of the ensembles learnt by the various algorithms at the points shown.

predictors increases, indicating that they are better able to utilize the information learnt. In particular, the curves for the RL approaches track much closer to the FE ones as compared to the CES and BP curves. These observations are stronger for the larger SS datasets (Fig. 4(b)) compared to the smaller PF ones (Fig. 4(a)). Furthermore, the substantial error bars on the CES curves indicate the method’s ad-hoc nature. These results show that our RL-based approaches are better able to utilize the information in larger datasets than CES to produce more accurate and stable predictions.

As emphasized earlier, a desirable characteristic of any ensemble selection method is the ability to discover/construct a parsimonious ensemble. To assess this ability of the ensemble selection algorithms we tested,

Table 2. Statistics for the curves shown for PF3 in Fig. 4(a). Column auESC shows the overall performance of the algorithm over all sizes of initial base predictors sets. The ratios of the size and performance of the produced ensembles to those of the best performing approach FE are shown at representative initial base predictors set sizes of 60, 120, and 180.

| | auESC | size_ratio@60 | size_ratio@120 | size_ratio@180 | perf_ratio@60 | perf_ratio@120 | perf_ratio@180 |
|----------------|--------|---------------|----------------|----------------|---------------|----------------|----------------|
| BP | 0.2654 | 0.0167 | 0.0083 | 0.0056 | 0.8919 | 0.9037 | 0.9005 |
| FE | 0.2974 | 1 | 1 | 1 | 1 | 1 | 1 |
| CES | 0.2816 | 0.31 | 0.24 | 0.19 | 0.9454 | 0.9493 | 0.9523 |
| RL_greedy | 0.2896 | 0.62 | 0.54 | 0.43 | 0.9615 | 0.9621 | 0.9746 |
| RL_pessimistic | 0.2869 | 0.37 | 0.24 | 0.19 | 0.9601 | 0.9531 | 0.9656 |
| RL_backtrack | 0.2893 | 0.57 | 0.52 | 0.48 | 0.9702 | 0.9714 | 0.9619 |

Table 3. Statistics for the curves shown for *A. thaliana* in Fig. 4(b). Column auESC shows the overall performance of the algorithm over all sizes of initial base predictors sets. The ratios of the size and performance of the produced ensembles to those of the best performing approach FE are shown at representative initial base predictors set sizes of 60, 120, and 180.

| | auESC | size_ratio@60 | size_ratio@120 | size_ratio@180 | perf_ratio@60 | perf_ratio@120 | perf_ratio@180 |
|----------------|--------|---------------|----------------|----------------|---------------|----------------|----------------|
| BP | 0.3833 | 0.0167 | 0.0083 | 0.0056 | 0.8118 | 0.7912 | 0.8237 |
| FE | 0.4769 | 1 | 1 | 1 | 1 | 1 | 1 |
| CES | 0.4549 | 0.40 | 0.31 | 0.24 | 0.9710 | 0.9577 | 0.9379 |
| RL_greedy | 0.4725 | 0.50 | 0.50 | 0.51 | 0.9946 | 0.9927 | 0.9945 |
| RL_pessimistic | 0.4634 | 0.48 | 0.29 | 0.21 | 0.9919 | 0.9649 | 0.9623 |
| RL_backtrack | 0.4721 | 0.87 | 0.79 | 0.75 | 0.9983 | 0.9919 | 0.9985 |

we show in Tables 2 and 3 important statistics of the curves shown in Fig. 4. Specifically, we compute ratios of the selected ensembles’ performance and sizes to those of the best performing and largest ensemble, FE. For brevity, we only show statistics at the representative initial base predictors set sizes of 60, 120, and 180.

From the results shown in Tables 2 and 3, it can be seen that CES discovers the smallest ensembles, but these appear not to have enough predictive information, resulting in lower classification performance than FE, especially in the case of the *A. thaliana* SS dataset. The RL-based approaches produce relatively larger ensembles due to their ability to search a much larger portion of the space of ensembles. Due to this same ability, they are able to select ensembles that produce classification performance nearly identical to FE, as shown by the performance ratios. Furthermore, the parsimony ability of these ensembles is enhanced as the number of initial base predictors increases, without significant variation in classification performance. This again validates the parsimonious yet accurate characteristic of the RL ensembles. Finally, our “pessimistic” strategy achieves the best parsimony-performance balance, possibly because of earlier resets of the learning episodes that force the agent to evaluate the upper levels of the lattice, where the smaller ensembles reside. We recommend analyzing summary statistics like the above to identify the ensembles representing the best parsimony-performance balance obtained from various ensemble selection methods. From another perspective, these ensembles might be identified as the point(s) at which the curve(s) like the ones in Fig. 4 start plateauing.

5.3. Dependence of ensemble selection algorithms’ behavior on parameters

The exploration probability ϵ is critical to the RL-based approaches as it controls the exploration/exploitation management, and consequently how much of the ensemble space is visited. To assess the effect of this parameter on the RL ensembles, we evaluated all three search strategies by executing them with $\epsilon \in \{0.01, 0.1, 0.25, 0.5\}$ for the PF datasets, and with $\epsilon \in \{0.01, 0.1, 0.25\}$ for the SS datasets. We then conducted ANOVA with the F-test to assess the effect of the ϵ values on both ensemble classification and size over the whole performance curves of the type shown in Fig. 4.

For the RL ensemble results from the PF3 dataset (Fig. 4(a)), both in terms of classification performance ($p = 0.26$) and ensemble size ($p = 0.75$), the RL_greedy approach produces similar results for different ϵ values. The RL_pessimistic and RL_backtrack strategies, however, produce statistically variable results with different epsilons, both in terms of classification performance ($p_{RL_pessimistic} = 7.3 \times 10^{-10}$, $p_{RL_backtrack} = 0.01$) and ensemble size ($p_{RL_pessimistic} = 2.7 \times 10^{-13}$, $p_{RL_backtrack} = 3.85 \times 10^{-4}$). The same trends are observed for PF1 and PF2. Further analysis of RL_pessimistic and RL_backtrack shows that as ϵ increases, the resultant ensemble performance drops, suggesting that too much exploration of the ensemble space may lead to overfitting.

The same analysis of the SS datasets also yielded similar results, with the exception that RL_backtrack did not show a dependence on the value of ϵ for any of the datasets, both in terms of classification performance (e.g., $p = 0.87$ for *A. thaliana*) and ensemble size (e.g., $p = 0.47$ for *A. thaliana*). Given the above observations, as well as the speed of execution, we show results only for $\epsilon = 0.01$ in the previous subsections.

Finally, as mentioned in Section 2.2, CES was run using the recommended values for its parameters.^{21,22} To test the impact of these parameters, which are generally difficult to set a priori, we vary their values as the pool size of candidate base predictors $m \in \{N/2, N\}$ and the maximum ensemble size $\in \{N/4, N/2, 3N/4, N\}$ ($N =$ total number of base predictors), and conducted a similar ANOVA analysis as above. The pool size did not have a significant impact on the performance, or the size of the ensembles ($p > 0.05$ for both PF3 and *A. thaliana*). However, the performance is significantly sensitive to the parameter controlling the maximum ensemble size ($p = 3.8 \times 10^{-3}$ for PF3 and $p = 2.8 \times 10^{-4}$ for *A. thaliana*), which also influences the size of the resulting ensemble slightly ($p = 0.08$ for PF3 and $p = 0.01$ for *A. thaliana*). Considering the above results with these, it can be inferred that CES is indeed more sensitive to its key parameters than the RL-based approaches, especially the best performing RL_greedy.

6. Discussion

Ensemble selection offers a powerful approach to addressing biomedical prediction problems, as well as gaining novel knowledge by the analysis of the selected ensembles. This paper presents a framework for selecting ensembles of classifiers using elements of reinforcement learning (RL), which offers a systematic and mathematically sound methodology for exploring the many possible combinations of base predictors that can be selected into an ensemble. This is in contrast to existing ensemble selection algorithms like CES, which often make ad-hoc decisions during ensemble learning and thus cannot offer performance guarantees. Several RL-based methods were implemented in our framework to search the space of possible ensembles as exhaustively as needed.

We tested our methods on two computational genomics problems, namely protein function prediction and splice site detection. We observed that our proposed RL-based methods are able to capture predictive performance close to the full ensembles with a much smaller number of base predictors. This observation is especially strong for the larger splice site datasets, an outcome worth investigating for other biomedical problems, such as cancer phenotype prediction. We also observed that many of the RL parameters, such as the exploration/exploitation probability (ϵ), did not have a significant impact on the downstream performance or sizes of the selected ensembles. It is necessary to examine the intrinsics of RL approaches, such as the effect of ϵ , on a variety of datasets to assess their strengths and weaknesses more comprehensively. Even more fundamentally, the RL approaches we tested can be reformulated, such as by revising the constituent reward functions, to yield better and/or more insightful ensembles. A particular avenue of interest here is to analyze the diversity of the base predictors selected into ensembles by the RL approaches in greater depth, and how that contributes to ensemble performance. Finally, there is also a need to investigate how the performance of our RL algorithms relates to the optimization capabilities and performance guarantees offered by Q-learning.

The RL algorithms studied allow for a larger portion of the space of possible ensembles to be examined more systematically, but this also causes computational requirements to grow substantially, especially as ϵ , the exploration probability, and the number of initial base predictors, grow. For instance, with our basic implementation of these algorithms, which are available as open-source software from <https://github.com/GauravPandeyLab/>, the time (on a 2.3 GHz processor) for one *A. thaliana* RL_greedy experiment varies from approximately one second with ten base predictors to approximately 3 minutes with 180 base predictors for $\epsilon = 0.01$. The same kind of executions took approximately 20 minutes on average for 180 base predictors with $\epsilon = 0.5$. For RL_pessimistic and RL_backtrack, the recorded times for an experiment with 180 base predictors and $\epsilon = 0.01$ are substantially higher, approximately 5.5 and 9.25 hours respectively, due to their more extensive exploration and resets in the search process. The memory requirements also vary similarly with the number of initial base predictors and value of ϵ . To make such executions more computationally feasible, especially for larger datasets, there is a need for developing more parallelized/optimized implementations of these algorithms. We will release such implementations in the future, and invite the community to participate in this effort.

To conclude, our overall effort was towards constructing accurate yet parsimonious (smaller) ensembles, which may in turn be more interpretable, for difficult problems. We acknowledge that interpretation can be a challenging and often subjective task, depending on the target problem, which is why it was out of the scope of our paper, and needs a deeper investigation. Although we didn't explicitly consider this, the interpretability of the base predictors itself would have a major impact on the interpretability of their resultant ensemble. Indeed, this interpretability criterion can be explicitly considered during the ensemble selection/search process to address this challenge more directly.

7. Acknowledgements

This work was partially supported by NIH grant # R01-GM114434 and an IBM faculty award to GP. We thank the Icahn Institute for Genomics and Multiscale Biology and the Minerva supercomputing team for their financial and technical support. We also thank Om P. Pandey and Gustavo Stolovitzky for their technical advice, and the anonymous reviewers for their comments and suggestions.

References

1. G. Pandey, V. Kumar and M. Steinbach, Computational Approaches for Protein Function Prediction: A Survey, Tech. Rep. 06-028, University of Minnesota (2006).
2. P. Radivojac, W. T. Clark, T. R. Oron et al., Nature methods **10**, 221 (2013).
3. M. Kuhn, M. Campillos, P. González, L. J. Jensen and P. Bork, FEBS letters **582**, 1283 (2008).
4. G. Schweikert, G. Rätsch, C. Widmer and B. Schölkopf, Adv. in Neural Info. Processing Systems **22**, 1433 (2009).
5. L. Rokach, Artificial Intelligence Review **33**, 1 (2009).
6. G. Seni and J. F. Elder, Synthesis Lectures on Data Mining and Knowledge Discovery **2**, 1 (2010).
7. P. Yang, Y. H. Yang, B. B. Zhou and A. Y. Zomaya, Current Bioinformatics **5**, 296 (2010).
8. A. Altmann, M. Rosen-Zvi, M. Prospero et al., PLoS ONE **3**, p. e3470 (2008).
9. A. Khan, A. Majid and T.-S. Choi, Amino Acids **38**, 347 (2010).
10. G. Pandey, B. Zhang, A. N. Chang et al., PLoS Computational Biology **6**, p. e1000928 (2010).
11. G. Yu, H. Rangwala, C. Domeniconi, G. Zhang and Z. Yu, Trans. on Comp. Biol. and Bioinfo. **10**, 1045 (2013).
12. Y. Guan, C. Myers, D. Hess et al., Genome Biology **9**, p. S3 (2008).
13. M. M. Ward, S. Pajevic, J. Dreyfuss and J. D. Malley, Arthritis Care & Research **55**, 74 (2006).
14. K. Tumer and J. Ghosh, Connection Science **8**, 385 (1996).
15. L. I. Kuncheva and C. J. Whitaker, Machine Learning **51**, 181 (2003).
16. T. G. Dietterich, Machine Learning **40**, 139 (2000).
17. R. E. Schapire and Y. Freund, Boosting: Foundations and Algorithms (MIT Press, 2012).
18. L. Breiman, Machine learning **45**, 5 (2001).
19. C. J. Merz, Machine Learning **36**, 33 (1999).
20. D. H. Wolpert, Neural Networks **5**, 241 (1992).
21. R. Caruana, A. Niculescu-Mizil, G. Crew and A. Ksikes, Intl. Conference on Machine Learning **21**, p. 18 (2004).
22. R. Caruana, A. Munson and A. Niculescu-Mizil, International Conference on Data Mining **6**, 828 (2006).
23. S. Whalen and G. Pandey, International Conference on Data Mining **13**, 807 (2013).
24. S. Whalen, O. P. Pandey and G. Pandey, Methods **93**, 92 (2016).
25. A. Niculescu-Mizil, C. Perlich, G. Swirszcz, V. Sindhwani and Y. Liu, J. of Mach. Learn. Research **7**, 23 (2009).
26. R. S. Sutton and A. G. Barto, Intro to Reinforcement Learning, 1st edn. (MIT Press, Cambridge, MA, USA, 1998).
27. T. R. Hughes, M. J. Marton, A. R. Jones et al., Cell **102**, 109 (2000).
28. C. L. Myers, D. R. Barrett, M. A. Hibbs, C. Huttenhower and O. G. Troyanskaya, BMC Genomics **7** (2006).
29. M. B. Shapiro and P. Senapathy, Nucleic acids research **15**, 7155 (1987).
30. G. Rätsch, S. Sonnenburg, J. Srinivasan et al., PLoS Comput Biol **3**, p. e20 (2007).
31. J. Kober, J. A. Bagnell and J. Peters, International Journal of Robotics Research **32**, 1238 (2013).
32. M. L. Puterman, Markov Decision Processes: Discrete Stochastic Dynamic Programming (1994).
33. C. J. C. H. Watkins, Learning from delayed rewards, PhD thesis, University of Cambridge England 1989.
34. C. J. C. H. Watkins and P. Dayan, Machine Learning **8**, 279 (May 1992).
35. D. P. Bertsekas and J. N. Tsitsiklis, Neuro-Dynamic Programming, 1st edn. (Athena Scientific, 1996).
36. E. R. Gansner and S. C. North, SOFTWARE - PRACTICE AND EXPERIENCE **30**, 1203 (2000).
37. M. Hall, E. Frank, G. Holmes et al., ACM SIGKDD Explorations Newsletter **11**, 10 (2009).
38. J. Demšar, Journal of Machine Learning Research **7**, 1 (2006).