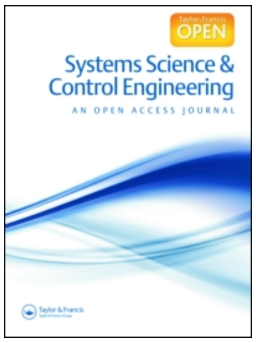

Systems Science & Control Engineering

An Open Access Journal

ISSN: (Print) 2164-2583 (Online) Journal homepage: https://www.tandfonline.com/loi/tssc20

An improved relief feature selection algorithm based on Monte-Carlo tree search

Jianyang Zheng, Hexing Zhu, Fangfang Chang & Yunlong Liu

To cite this article: Jianyang Zheng, Hexing Zhu, Fangfang Chang & Yunlong Liu (2019) An improved relief feature selection algorithm based on Monte-Carlo tree search, Systems Science & Control Engineering, 7:1, 304-310, DOI: 10.1080/21642583.2019.1661312

To link to this article: https://doi.org/10.1080/21642583.2019.1661312

SYSTEMS SCIENCE & CONTROL ENGINEERING: AN OPEN ACCESS JOURNAL
2019, VOL. 7, NO. 1, 304–310
https://doi.org/10.1080/21642583.2019.1661312

Taylor & Francis
Taylor & Francis Group

OPEN ACCESS Check for updates

An improved relief feature selection algorithm based on Monte-Carlo tree search

Jianyang Zheng, Hexing Zhu, Fangfang Chang and Yunlong Liu

Department of Automation, Xiamen University, Xiamen, People's Republic of China

ABSTRACT
The goal of feature selection methods is to find the optimal feature subset by eliminating irrelevant or redundant information from the original feature space according to some evaluation criteria. In the literature, the Relief algorithm is a typical feature selection method, which is simple and easy to execute. However, the classification accuracy of the Relief algorithm is usually affected by the noise. In recent years, the Monte Carlo Tree Search (MCTS) technique has achieved great success in strategy selections of large-scale systems by building a tree and quickly focusing on the most valuable part of the search space. In this paper, with the benefit of MCTS, an MCTS-based feature selection approach is proposed to deal with the feature selection problem of high dimensional data, where the Relief algorithm is used as the evaluation function of the MCTS approach. The effectiveness of the proposed approach is demonstrated by experiments on some benchmark problems.

ARTICLE HISTORY
Received 25 February 2019
Accepted 26 August 2019

KEYWORDS
Feature selection; Relief algorithm; Monte-Carlo tree search

Introduction

While the high-dimensional data is ubiquitous in science and engineering, e.g. multimedia data, gene expression profile data, etc., it usually contains a large amount of irrelevant or redundant information. To eliminate such irrelevant or redundant information, the feature reduction techniques that can remove irrelevant features while remain feature subsets with high classification performance are usually needed (Hu, 2008; Wang, Li, & Fang, 2012).

In the literature, filter, wrapper, and embedded methods are the three main techniques for feature selection. For the filter related methods, prior to modelling, they use a 'proxy measure' calculated from the general characteristics of the training data to score features or feature subsets. Filter methods are generally much faster and function independently of the induction algorithm, meaning that the selected features can then be passed to any modelling algorithms (Urbanowicz, Melissa, La, Olson, & Moore, 2018). Wrapper methods employ any stand-alone modelling algorithm to train a predictive model by using a candidate feature subset, where the testing performance on a hold-out set is typically used to score the feature set. In any wrapper methods, a new model must be trained to test any subsequent feature subsets, therefore wrapper methods are typically iterative and computationally intensive, but the related techniques can identify the best performing features set for one specific modelling algorithm (Urbanowicz et al., 2018). For the embedded methods, they perform feature selection as a part of the modelling algorithm's execution, which tend to be more computationally efficient than wrapper methods as they can simultaneously integrate modelling with feature selection (Urbanowicz et al., 2018). For all these techniques, the filter methods have advantages of low computational complexity and high efficiency compared to wrapper methods, and is suitable for processing large-scale data. In addition, different from the embedded methods, the filter methods are independent of the modelling algorithm and have better versatility.

The Relief algorithm (Kira & Rendell, 1992; Kononenko, 1994) is a typical filter method, which considers the correlation between features and uses feature weights as the basis for selecting classification features. Although the calculation of the classification weights by the Relief algorithm is simple, the results are easily affected by the noise, which may lead to inaccuracies in the obtained subset of features.

The Monte Carlo Tree Search (MCTS) (Browne et al., 2012) algorithm is a search algorithm that combines the traditional Monte Carlo random sampling method and tree search method. As the MCTS algorithm can quickly focus on the most valuable part of the search space, it has achieved great success in the strategy selection of large-scale systems, for example, MCTS based Alpha Go has defeated the human Go champion (Silver et al., 2016). The MCTS approach searches the solution by building an

CONTACT Yunlong Liu ylliu@xmu.edu.cn

asymmetric tree iteratively (Liu, Zhu, Zeng, & Dai, 2016). With the increase of the iterations, in the search tree, the node with the smaller evaluation value will gradually be ignored in the process of expanding the tree, while the nodes with the larger evaluation value have more chances to expand a child node. By selecting a suitable evaluation function, for the feature selection problem, the node with the small evaluation value will usually correspond to the noise information, and then such noise information can be effectively removed in the asymmetric construction process of the MCTS approach.

In this paper, with the benefits of MCTS for finding effective solutions in large scale systems, we propose an MCTS-based feature selection approach by adopting the Relief algorithm as the evaluation function. Because the MCTS algorithm is originally used to solve sequential decision problems and the evaluation function is used to evaluate the value of the selected action sequence (Browne et al., 2012), they cannot be directly applied to the feature selection problem. In our approach, we firstly transform the feature selection problem into a sequential decision problem, then, the Relief algorithm is used as the evaluation function of the MCTS, and a search tree on the features is iteratively built, finally, the possible best feature subset is selected according to the search tree.

The rest of this paper is structured as follows: A brief introduction to the Relief algorithm and Monte-Carlo tree search is discussed in Section Background. The next section presents the framework of our MCTS-based feature selection approach. Experimental results from several standard problem domains are evaluated and compared in Section Experiments. Finally, we conclude and highlight some possibilities for future work.

Background

Relief algorithm

The Relief algorithm is efficient for selecting the relevance features, whose key idea is to select features according to how well their values distinguish among instances that are near each other (Kira & Rendell, 1992; Kononenko, 1994). For a given instance, Relief searches for its two nearest neighbours: one from the same class called nearest hit and the other from different class called nearest miss (Kononenko, 1994). Relief estimates the weight W of feature A, i.e. $W[A]$, as follows:

$$\begin{aligned} W[A] = {} & P(\text{different value of } A| \\ & \quad \text{nearest instance from different class}) \\ & - P(\text{different value of } A| \\ & \quad \text{nearest instance from the same class}) \end{aligned} \tag{1}$$

where P is the probability of different values of A with different instances. The above calculation follows that good features should differentiate between instances of different classes and have the same value for instances in the same class.

The Relief algorithm is shown in Algorithm 1, where function *diff* calculates the difference between the values of feature for two instances, parameter m represents the number of instances for approximating the probabilities, and a larger m means a more reliable approximation. The obvious choice for relatively small number of training instances is to set m to the upper bound and run the outer loop of the learning algorithms over all available training instances (Kononenko, 1994). For discrete features the difference is either 1 (the values are different) or 0 (the values are equal), while for continuous features the difference is the actual difference normalized to the interval $[0, 1]$, then, normalization with m guarantees that all weights are in the interval $[-1, 1]$ (Kononenko, 1994). Function *diff* is also used to calculate the distance between instances to find the nearest neighbours and the total distance is the sum of differences over all features. Obviously, the Relief algorithm tries to approximate the result of Equation (1).

Algorithm 1: Relief Algorithm

```
1: set all weights $W[A] \leftarrow 0$
2: for $i = 1 \sim m$ do
3:   randomly select an instance $R$
4:   find nearest hit $H$ and nearest miss $M$
5:   for $A = 1 \sim$ all features do
6:     $W[A] = W[A] - diff(A, R, H)/m + diff(A, R, M)/m$
7:   end for
8: end for
```

As can be seen from Algorithm 1, it is important to choose the nearest neighbours in Relief. However, as redundant and noisy features may affect the selection of nearest neighbours, the estimation of feature weight becomes unreliable. To increase the reliability of the feature weight, the Relief algorithm can be extended to search for the k-nearest hits/misses instead of only one nearest hit/miss. The extended version of the RELIEF algorithm, called RELIEF-A, averages the contribution of k-nearest hits/misses (Kononenko, 1994), which is shown in Equation (2):

$$\begin{aligned} W[A] = W[A] - \sum_{j=1}^{k} diff(A, R, H_j)/(mk) \\ + \sum_{j=1}^{k} diff(A, R, M_j)/(mk) \end{aligned} \tag{2}$$

As mentioned previously, the Relief algorithm measures the classification information contained in each feature by calculating feature weights, then the nearest neighbours can be searched by calculating the distance between the instances, and in the process, all the features are considered. However, in practice, there may be only a few features contain classification information, and the other features exist as noise and have no direct relationship with classification (Wang et al., 2012). The data sets with high dimensions are ubiquitous in real-world applications and with the increase of the data dimension, lots of irrelevant and redundant information may exist in these data, which has leaded to the problems of 'Curse of Dimensionality' and 'Over-Fitting' in the machine learning literature. For example, in the dataset of the gene expression profiles, the dimension of the data is several thousand, but the information that is really useful is usually only tens of dimensions, and the work of [1] shows that only tens of dimensions of the data contained the most relevance information.

Monte Carlo tree search

Monte Carlo Tree Search (MCTS) is a method for finding the optimal decisions in a given domain by taking random samples in the decision space and building a search tree according to the results of samples. The basic process of MCTS is to construct the search tree iteratively until the pre-defined termination condition is reached (Liu et al., 2016). The node in the search tree represents the state of the domain, and the directed links from a node to its child nodes represent actions leading to subsequent states (Browne et al., 2012). Four phases exist in each search iteration (Hunt, Marin, & Stone, 1966; Chaslot, Bakkes, Szita, & Spronck, 2008).

Selection: Starting from the root node, a child selection policy is applied recursively to go down the tree until the most urgent extensible node is reached. A node is extensible if it represents a nonterminal state and has unvisited children.

Expand: According to the available actions, one (or more) child nodes are added to expand the tree.

Simulation: Run a simulation from the new node(s) according to the default strategy to produce an outcome.

Backpropagation: The simulation result is 'backed up' through the selected nodes to update their statistics.

These may be grouped into two distinct policies (Browne et al., 2012): One is the tree policy, i.e. select or create a leaf node from the nodes already contained within the search tree (selection and expansion); the other is default policy, i.e. play out the domain from a given nonterminal state to produce a value estimate (simulation).

Feature selection with Monte Carlo tree search

Transformation of feature selection into sequential decision

As mentioned above, the MCTS approach has achieved great success in many large-scale applications. Inspired by the earlier work in (Gaudel & Sebag, 2010), we attempted to improve the performance of the Relief algorithm by combining the MCTS technique with the Relief algorithm. In this section, we show how the MCTS technique can be used to solve the feature selection problem of high dimensional data. As the MCTS algorithm is originally designed to find the best strategy for sequential decision making problems, which is obviously different from the problem of feature selection. In our approach, we first transform the feature selection problem into a sequential decision problem, where when constructing the search tree, each node in the tree represents a feature, the feature set from the root node to the current node is treated as the current state, and the set of features is used as the action set. Then, the feature selection problem is transformed into a sequential decision making problem.

As in the MCTS approach, an evaluation function is required to guide the search to a better path. In our MCTS-based feature selection approach, to evaluate the contribution of the selected features to the classification, with the advantages of the Relief algorithm, i.e. simple, low computational complexity, etc., it is used as the evaluation function for each feature set (Note that the Relief algorithm described later refers to the Relief-A algorithm). Using the Relief algorithm to calculate feature weights of all features in the feature set, we can get a feature weight vector:

$$W = [W(1), W(2), \ldots W(m), \ldots, W(n)] \tag{3}$$

where n is the number of features, and m is the number of features that have been selected by the MCTS algorithm before the starting of the simulation phase. Each feature weight $W(i)$ in the vector can be calculated by Equation (2).

The optimal strategy π^* obtained from the MCTS algorithm is a feature set that maximizes the sum of feature weights:

$$\pi^* = \arg\max_{\pi} \sum_{i=1}^{m} W(\pi_i) \tag{4}$$

Different from the true decision making problem, in this feature selection problem, the same actions are not allowed to appear in one sequence, i.e. the two action (feature) sets cannot be identical, and it is not allowed to have the same actions (features) in one action (feature) set.

MCTS-Relief algorithm

By transforming the feature selection problem into a sequential decision problem and using the Relief-A algorithm that calculates feature weights as the MCTS evaluation function, the MCTS approach can be used for feature selection. This MCTS-based feature selection algorithm is called the MCTS-Relief algorithm, and just as the MCTS approach, four phases are contained in the algorithm:

Selection: Starting from the root node, which is an empty set initially, the actions (features) are selected strategically one by one according to the UCB algorithm (Kocsis & Szepesvári, 2006):

$$a^* = \arg\max_a (Q(s,a) + c\sqrt{\frac{\log N(s)}{N(s,a)}}) \tag{5}$$

where a^* is the action(feature) corresponding to the child node with the largest UCB value under the current node, $Q(s,a)$ the value of the current node when selecting the corresponding action(feature), $N(s)$ the number of times the current node is selected, $N(s,a)$ the number of times the corresponding child action of the current node is selected respectively, and $c > 0$ is a tuning constant. There is an essential balance between the first (exploitation) and second (exploration) terms of the UCB equation. As each node is visited, the denominator of the exploration term increases, which decreases its contribution. On the other hand, if another child of the parent node is visited, the numerator increases and hence the exploration values of unvisited siblings increase. The constant c in the exploration term can be adjusted to lower or increase the amount of exploration performed (Browne et al., 2012). With the above UCB algorithm, in the feature selection, the features with large classification weight will be taken into account, and with the increase of the number of iterations, the search tree grows asymmetrically, features with small classification weight (usually noise features) will be filtered out.

Expand: According to Equation (5), we can see that if a candidate action a(feature) has not been ever selected at current node s (candidate action (feature) cannot be included in the set of features selected from the root node to current step), where $N(s,a) = 0$, it should be expanded under the current node. However, as mentioned previously, redundant information may exist between features, when selecting which feature should be expanded, the relevance degree between the candidate feature and the current state should be taken into account. In our approach, Pearson correlation coefficient is used to measure the correlation between features. Suppose there are two features, vectors A and B represent the values of the two features in each sample, then the calculation of Pearson correlation coefficient is as follows (Benesty et al., 2009):

$$r = \frac{\sum_{i=1}^{n}(A_i - \bar{A})(B_i - \bar{B})}{\sqrt{\sum_{i=1}^{n}(A_i - \bar{A})^2}\sqrt{\sum_{i=1}^{n}(B_i - \bar{B})^2}} \tag{6}$$

As the larger the correlation coefficient, the stronger the correlation is, in our approach, a correlation coefficient threshold Cp is set, and only the feature that the correlation coefficient between it and the feature represented by the current node is smaller than Cp is expandable. As shown in earlier work, the selection of new nodes can be benefitted from prior knowledge (Gelly & Silver, 2007). We set some node extension rules to prevent hopeless exploration of feature space. The candidate feature sets are sorted according to the feature weights calculated by the original feature set, and the first few features after the feature represented by the current node are selected as the extended nodes. This prevents the expansion of meaningless nodes and makes the selection somewhat random, and ensures that the selected feature set for each path will not be identical.

Simulation: After the expansion of the current node, the simulation phase starts where actions (features) are randomly selected until some predefined conditions are reached. Then, the Relief algorithm is used to calculate the feature weight vector W of the feature set which contains all the actions (features) that are selected from the root node to the end of this simulation. The first m weights in W(m is the number of features selected in the selection and expansion phase) are used as the *reward* of the selected feature set, i.e. $reward = [W(1), W(2), \ldots, W(m)]$.

Backpropagation: In the backpropagation stage, the *reward* obtained in the simulation stage is used to update each node s visited during the search and expand phase, and the value of node s is updated as follows:

$$Q(s) \leftarrow Q(s) + \frac{r - Q(s)}{N(s,a)} \tag{7}$$

where $Q(s)$ is the value of the node s, $Q(s)$ is initialized to zero for each s, and r is the weight of s, i.e. the s^{th} element of *reward*. $N(s)$ and $N(s,a)$ are also updated.

After constructing the search tree, starting from the root node, the features with the largest value are selected and used as the selected feature set.

Experiments

Experimental datasets

Four benchmark datasets of binary classification problem are chosen for the experiments: Prostate (Zhang, Hu, Li, & Zhang, 2014; Singh et al., 2002), The Central Nervous System (CNS) (Pomeroy et al., 2002), Leukemia (Zhang et al., 2014), and Colon (Zhang et al., 2014). Firstly, the genes (features) of these datasets were selected, and then the selected genes were used for classification to verify the effect of the proposed algorithm. The details of these datasets are shown in Table 1.

Experimental setting

For our proposed MCTS-Relief algorithm, the number of candidate features for each node is set to 10. The search constant *c* in Equation (5) is set to 0.1. In order to eliminate the features with redundant information and increase the search efficiency, the feature correlation threshold *Cp* is set to 0.9. The search depth is set to 20, and to verify the impact of different number of extended nodes on the algorithm, the number of extended nodes is set to 5 and 10 respectively. For the number of simulations to build a search tree, it is set to 1000 (If the required number of features cannot be obtained in one built tree, a new tree is built by using the obtained set of features in the previous built tree as the root node). We compared the MCTS-Relief algorithm with the Recursive Feature Elimination Relief (RFE-Relief) (Li, Li, & Ruan, 2006; Yu, 2012) algorithm. The basic idea of the RFE-Relief algorithm is as follows: Firstly, the Relief algorithm is used to calculate the feature weights for all the features in the feature set. Then after removing 10% of the features with the small weights, the Relief algorithm is continued to be used to calculate the classification weights of the left features, and then the 10% of the features with the small weights is removed again. The above process is executed recursively. It can be seen that the RFE-Relief algorithm can reduce the features that have no effect on classification, and therefore the impact of the noise continues to decrease. The RFE-Relief algorithm can be briefly described as in Algorithm 2 (Li et al., 2006; Yu, 2012).

Performance measurement

Assume that the classification feature set selected by feature selection is F_n, and Support Vector Machine (SVM)

Table 1. Experimental datasets.

Name	Prostate	CNS	Leukemia	Colon
number of features	12534	7129	5147	2000
number of samples	102	60	72	62

Algorithm 2: RFE-Relief Algorithm

```
1: $F = \{g_1, g_2, \ldots, g_n\} \leftarrow$ initial feature set
2: while the number of features in $F$ is greater than the preset number do
3:   $W = RELIEF(F) \leftarrow$ weights $W$ of $F$
4:   $c = \arg\min W \leftarrow$ features with small weights
5:   $F = F - F(c) \leftarrow$ new feature set
6: end
```

(Cherkassky & Mulier, 1998) is used to measure the classification ability of F_n. Consider one has the following training data set:

$$S_t = \{(x_i, y_i) | x_i \in R^d, y_i \in \{-1, 1\}, i = 1, 2, \ldots, n\}$$

An SVM is a classifier constructed from sums of the kernel functions:

$$g(x) = \operatorname{sgn}\left(\sum_{i=1}^{n} \alpha_i y_i K(x, x_i) + b\right) \tag{8}$$

where *g(x)* is the discriminant function of SVM, $K(x, y)$ is a kernel function, and in this paper, the RBF kernel function is used.

Experimental result and analysis

To evaluate the classification performance of the obtained feature set, we use *n*-fold cross-validation (CV accuracy) for the evaluation. Divide the original dataset into *n*, and make each subset as the test set once, and the remaining *n-1* subset as the training set. The MCTS-Relief algorithm (Note that the MCTS-Relief algorithm with the *k* extended nodes is called MCTS-Relief-*k*) and the RFE-Relief algorithm were used to select features on the mentioned four experimental datasets. Then the SVM classifier was used to execute the *n*-fold cross-validation to test the classification abilities of the genes selected by the two methods, where *n* was set to 10. Since the results obtained by the MCTS-Relief algorithm each time are somewhat different, we run 5 rounds of experiments on it and average the results. The classification results of the four datasets are shown in Figures 1–4 respectively, where *x*-axis is the number of the selected genes and *y*-axis is the classification accuracy.

Firstly, we tested the impact of different extended nodes on the performance of our proposed technique. As shown in the experimental results, in the Prostate dataset, when the number of extended nodes is set to 5 and 10, the algorithm works similarly. In other datasets, if the number of extended nodes is 5, the performance of the algorithm will be slightly worse than the case where the number of extended nodes is 10. It can also be seen that the number of extended nodes must be kept at a certain number to ensure the performance of the algorithm. For

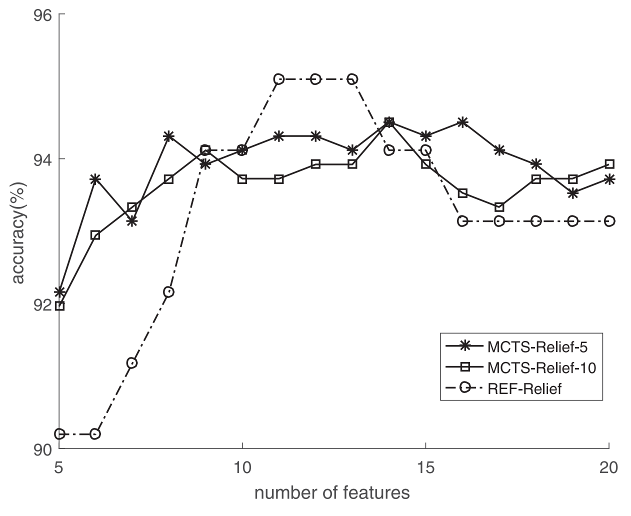

Figure 1. Prostate.

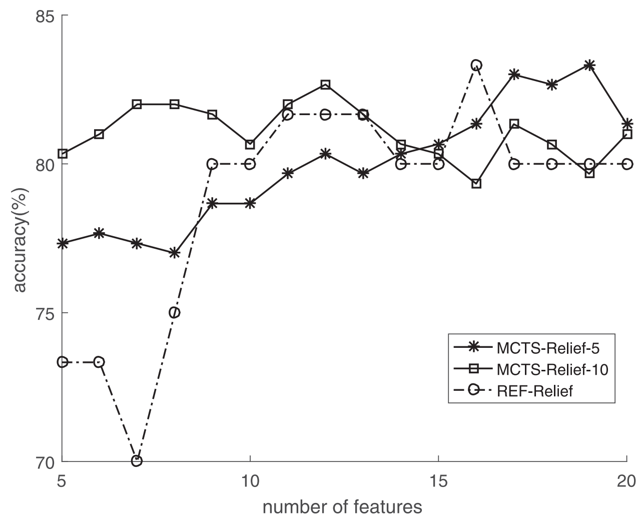

Figure 2. CNS.

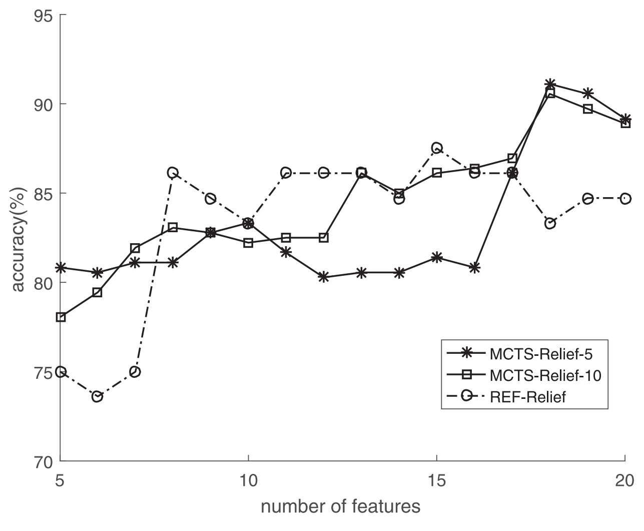

Figure 3. Leukemia.

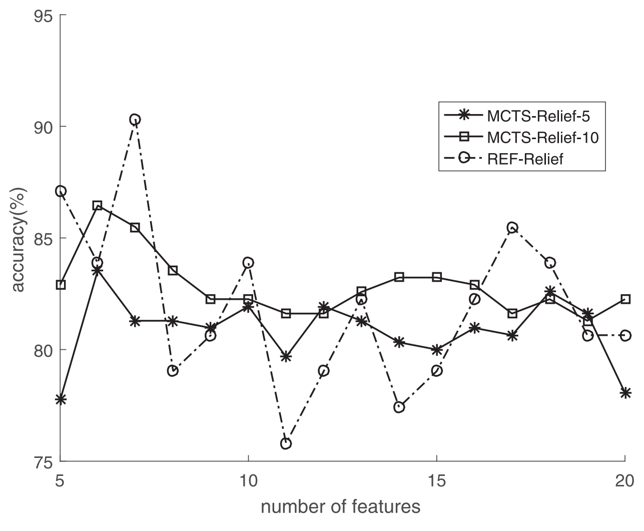

Figure 4. Colon.

the datasets used in the experiment, it is more suitable to set the number of the expansion nodes to 10 where better results can be obtained.

Then, we compared the performance of the MCTS-Relief-10 algorithm with the REF-Relief algorithm. As can be seen from Figures 1–4, for the Prostate and CNS dataset, in most of the cases, the MCTS-Relief algorithm achieved a better performance compared to the REF-Relief algorithm, and for the Leukemia dataset and Colon dataset, the MCTS-Relief algorithm is still competitive. In the case that fewer features were selected, in all datasets, the MCTS-Relief algorithm is more stable than the RFE-Relief algorithm. This phenomenon may be due to the fact that as the number of iterations of the RFE-Relief algorithm increases, fewer features remain. When the number of features is too small, some important features are eliminated, which may also lead to inaccurate results. However, the MCTS-Relief algorithm always uses a certain number of high-weight features for each iteration calculation, so that better results are obtained when the selection features are small. We can also see from the results that with the increase of the number of selected features, these two algorithms achieve the nearly same performance, the possible explanation is that as the number of selected features increases, the classification information contained in the selected feature sets tends to be complete.

The experimental results demonstrate the possibility of the application of MCTS for feature selection of high dimensional data by building an asymmetry search tree, where with the increase of the number of simulations and in the process of expanding the search tree, the features with the small classification weight is eliminated, which are usually the noise features.

Conclusion and future work

In this paper, we show the possibility of the application of MCTS for feature selection by transforming the feature selection problem into a sequential decision problem, and an MCTS-based feature selection approach is proposed, where the Relief algorithm to calculate feature weights is used as the evaluation function. The efficiency of the proposed approach is demonstrated by the results of four benchmark problems. Future work includes the application of the MCTS algorithm to deal with multi-class sample feature selection problems and the combination of the MCTS algorithm with other domain classical algorithms to handle the huge search space problem that is difficult to solve by existing methods.

Disclosure statement

No potential conflict of interest was reported by the authors.

Data availability statement

The data that support the findings of this study are available from the corresponding author, upon reasonable request.

Funding

This work is supported by the National Natural Science Foundation of China [grant numbers 61772438 and 61375077].

ORCID

Jianyang Zheng http://orcid.org/0000-0002-4548-4753
Yunlong Liu http://orcid.org/0000-0001-8448-7845

References

Benesty, J, Chen, J, Huang, Y, & Cohen, I. (2009). *Pearson Correlation Coefficient. In Noise Reduction in Speech Processing.* (Vol. 2). Berlin, Heidelberg: Springer.

Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., . . . Colton, S. (2012). A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, *4*(1), 1–43.

Chaslot, G., Bakkes, S., Szita, I., & Spronck, P. (2008). Monte-Carlo tree search: A new framework for game AI. In *AIIDE*.

Cherkassky, V., & Mulier, F. (1998). Statistical learning theory. *Encyclopedia of the Sciences of Learning*, *41*(4), 3185–3185.

Gaudel, R., & Sebag, M. (2010). Feature selection as a one-player game. In *International Conference on machine learning* (pp. 359–366).

Gelly, S., & Silver, D. (2007). Combining online and offline knowledge in UCT. In *Proceedings of the 24th international conference on machine learning* (pp. 273–280). ACM.

Hu, J. (2008). Survey on feature dimension reduction for high-dimensional data. *Application Research of Computers*, *25*(9), 2601–2606.

Hunt, E. B., Marin, J., & Stone, P. J. (1966). Experiments in induction. *American Journal of Psychology*, *80*(4), 17–19.

Kira, K., & Rendell, L. A. (1992). The feature selection problem: Traditional methods and a new algorithm. In Aaai (Vol. 2, pp. 129–134).

Kocsis, L., & Szepesvári, C. (2006). Bandit based monte-carlo planning. In *European conference on machine learning* (pp. 282–293). Springer, Berlin, Heidelberg.

Kononenko, I. (1994). Estimating attributes: Analysis and extensions of RELIEF. In European conference on machine learning (pp. 171–182). Springer, Berlin, Heidelberg.

Li, Y. X., Li, J. G., & Ruan, X. G. (2006). Study of informative gene selection for tissue classification based on tumor gene expression profiles. *Chinese Journal of Computers-Chinese Edition*, *29*(2), 324.

Liu, Y., Zhu, H., Zeng, Y., & Dai, Z. (2016). Learning predictive state representations via monte-carlo tree search. Proceeding of the 25th International Joint Conference on Artificial Intelligence (IJCAI-16), (pp. 3192–3198).

Pomeroy, S. L., Tamayo, P., Gaasenbeek, M., Sturla, L. M., Angelo, M., McLaughlin, M. E., . . . Allen, J. C. (2002). Prediction of central nervous system embryonal tumour outcome based on gene expression. *Nature*, *415*(24), 436–442.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., . . . Dieleman, S. (2016). . Mastering the game of go with deep neural networks and tree search. *Nature*, *529*(7587), 484–489.

Singh, D., Febbo, P. G., Ross, K., Jackson, D. G., Manola, J., Ladd, C., . . . Lander, E. S. (2002). Gene expression correlates of clinical prostate cancer behavior. *Cancer Cell*, *1*(2), 203–209.

Urbanowicz, R. J., Meeker, M., Cava, W. L., Olson, R. S., & Moore, J. H. (2018). Relief-based feature selection: Introduction and review. *Journal of Biomedical Informatics*, *85*, 189–203.

Wang, S. L., Li, X. L., & Fang, J. (2012). Finding minimum gene subsets with heuristic breadth-first search algorithm for robust tumor classification. *BMC Bioinformatics*, *13*(1), 178.

Yu, L. (2012). Study on selection of characteristic gene of colon cancer based on improved relief Algorithm. Naikai University.

Zhang, J., Hu, X. G., Li, P. P., & Zhang, Y. H. (2014). Informative gene selection for tumor classification based on iterative lasso. *Pattern Recognition and Artificial Intelligence*, *27*(1), 49–59.