PAFS – An Efficient Method for Classifier-Specific Feature Selection

Pham Quang Huy
Department of Mathematics and Computer Science
University of Dalat, Dalat, Vietnam.
School of Computer Science, University of Windsor
Windsor, N9B 3P4, Ontario, Canada.

Alioune Ngom, Luis Rueda
School of Computer Science
University of Windsor
Windsor, N9B 3P4, Ontario, Canada.

***Abstract*— An optimal classification model for classifying on a given problem should comprise of a classifier, a proper feature subset and a parameter set such that the classifier can attain high prediction performance as possible. Many recent feature selection methods are either too exhaustive or too greedy. Besides, many classification approaches conduct parameter search after feature selection stage, resulting in the classification results that are not as optimal as they should. In this study, we propose a new greedy selection method, called *Parallel Apriori-like Feature Selection* (PAFS), which searches for an optimal classification model in the combined space of features and parameters. Moreover, its greedy search behavior is controllable by running options so that it is flexible for different problems. We also devised a *Tree-based Classifier Model* (TCM) algorithm which wraps PAFS in solving multi-class problems. Our methods achieved excellent results when applied on two multi-class datasets. In particular, on a breast cancer dataset consisting of 5 classes and 13582 features, our methods selected feature subsets of no more than 10 features and each with the prediction accuracy of at least 94%.**

Keywords*— *Feature selection, grid search, Parallel Apriori-like Feature Selection, PAFS, Tree-based Model for Multi-class Problem, optimal classification model search.

I. INTRODUCTION

In reality, many machine learning tasks involve high dimensional data which contain thousands of features and millions of samples. For example, a breast cancer dataset in [1] has up to 13582 features, where each feature is a gene expression; some text classification problems involve thousands of m-grams. The curse of dimensionality might keep many powerful learning techniques such as Support Vector Machine (SVM) [2, 3, 4], Bayesian Network [5], K-Nearest Neighbors (K-NN) [2], Decision Tree [6] from being applicable in those cases. In such data, there are usually a lot of redundant and irrelevant features that can be removed. Thus, feature selection is one of the best remedies. This pre-processing technique is to find the most informative subset of features which can still generalize the original data. As a consequence, the training time on the projected data can be significantly reduced, the result can be explained easier, and the prediction accuracy can be improved. Feature selection is especially useful in domains where there are much more features than samples. Such domains include written text analysis, bioinformatics classification/prediction, where there are many thousands of features but only a few tens to hundreds of samples. It is shown in [1, 7, 8, 9,] that, in many situations, only a small number among thousands of features strongly correlate with the target feature. However, extracting useful features from a universal set of hundreds or thousands of features is really challenging as the search space is exponential to the number of features. Exhaustive search is almost impossible; level-wise search must base on reasonable some stop conditions for a good quality; greedy search like Greedy Forward Selection might find only a local optimum. Therefore, in some cases, it requires a more suitable strategy to guide the search.

Besides, to obtain a further improvement for some learners, such as SVM, K-NN, there must be an optimal parameter set. While the parameter space still can be continuous, trying to find the global optimum is almost impossible. Grid search is a common and straightforward way to find an optimal parameter setting by sparsely sweeping through the parameter space. Moreover, the classification performance is different from one combination of parameters and feature subset to another. Thus, searching for optimal parameter after feature selection step probably miss the solution that should be chosen (i.e., when combining a feature subset that was skipped in feature selection stage with a suitable parameter setting, we may have better classification performance). Hence, it seems a good idea to optimize both the subset of features and the parameters by searching in their combination space.

Finally, a classifier can be suitable for one problem (dataset) but might not for another. Thus, given a dataset, the optimal classification model must comprise of a classifier, the most informative feature subset and the optimal parameters for that classifier.

Our contribution. These observations motivate us to propose a new method, called Parallel Apriori-like Feature Selection (PAFS), that finds the optimal classification model for a given classifier by exploring the features space and the parameter space at the same time. The ideas of combining the spaces may not be new, the way PAFS search for the feature subset is. Its search behavior is oriented to the optimal results and is controllable by running options to avoid being too exhaustive or too greedy. This algorithm is mainly to cope with binary classification problems. To handle multi-class problems,

we extend the tree-based model in [1] to another algorithm called Tree-based Classification Model for Multi-class Problem (TCM) which wraps PAFS and integrates the space of classifiers as well. It outputs a single-path tree where each node is a binary classification model learned by PAFS. Both PAFS and TCM can be implemented in parallel. When classifying, each binary model is applied, according to its order priority, to identify whether a new instance belongs to the class of that node or not.

When applied for classifying on a breast cancer of five subtypes, our model returned accuracy of about 99%; about 5% more than that reported in [1] and required fewer genes. On an erythemato-squamous disease dataset, we obtained the similarly high accuracy.

The application of our approach is not limited to classification problems, but it can be extended to regression problems as well.

Paper organization. The rest of the paper is organized as follows. In section II, we briefly summary the feature selection approaches, and some related works on feature selection and on classifying breast cancer subtypes. Details of our proposed algorithms are presented in section III, and the behavior and complexity of PAFS are analyzed as well. Information about datasets, implementation notes, the experiment results, and discussion are described in section IV. Finally, in section V, we summary our contribution and point out what should be investigated further.

II. RELATED WORKS

In this section, we first summary the feature selection approaches, and then highlight some recent works on feature selection and on breast cancer subtype classification that related the most to this study.

Generally, feature selection methods are categorized into the following approaches.

Filter approaches. Filter approaches can be considered as selecting the top features providing the most information about the classes, based on a particular statistic criterion. Some of the popular criteria are Information Gain, Gain Ratio [6], Chi-Square [1], mRMR [7, 8]. Since the features are often evaluated independently from each other, these approaches are clearly fast. They are effective to shrink the features space, especially when the number of features in the dataset is large. However, they might not perfectly eliminate redundancy because the presence of one feature may reduce the impact of some others on the class feature. And, as they are not tailored to any specific classifier, in many cases, the selected features are used as input for another processing steps rather than as the final feature subset for classification [9, 10].

Wrapper approaches. Wrapper methods, on the other hand, consider a subset of features at a time and search for the optimal feature subset with regard to a specific classifier. Thus, they somehow take the between-feature dependencies into account. For each subset, the target classifier is trained and tested on the projected data to score the fitness. Various strategies can be applied to traverse the space of feature subsets, including exhaustive search and greedy searching methods such as Greedy Forward Selection, Greedy Backward Elimination, Floating With Forward Selection, Floating With Backward Selection are likely preferred. In greedy methods, starting from a subset of features, new candidates are gradually generated by adding/deleting a feature and then evaluated. Only one or two best ones are maintained to generate candidates for the next step and so on. The process stops when some conditions are met, e.g., the size of subset reaches a certain threshold or the decline of quality excesses a threshold. Basically, none of them guarantees to find the global optimum.

Wrapper methods usually produce better performance than filter ones, but they require expensive computations, especially, when classifiers of high computational cost such as SVM or Bayesian Network are chosen.

Hybrid approaches. They are the combinations of filter and wrapper methods. Candidate features are first selected by a filter criterion to prune the feature search space before a wrapper method is applied to find the final subset. One can see that they can take advantages and reduce the drawbacks of the two previous approaches.

Embedded approaches. In these approaches, the search for an optimal feature subset is integrated with the process of constructing the classifier. Decision tree learning can be considered an instance of this method.

Based on one-against-all strategy, the authors of [11] applied an ensemble learning approach to deal with the multi-class problem. The classification problem of c classes is first transformed into c binary classification problems. In each subproblem, the *ith* class is considered as the positive class while the others are combined as the negative one. Then, they apply traditional feature selection for each subproblem. In classification, all classifiers of the subproblems vote for the class label of the new instance. To cope with the imbalanced data, the class of the minority can be oversampled. They tested their approach with Naïve Bayes, K-NN, C4.5 on 15 datasets, whose number of features range from 4 to 64. Three of the datasets have more the 3000 samples. On average, their approach was better than the traditional approaches from 2% to 3% of accuracy.

In [1], also based one-against-all strategy, the authors introduced a tree-based model for classification. Its main idea is to transform the multi-class problems into binary problems, but the easiest one is chosen to build the current node. Then, after removing the samples of the chosen class, the remaining data are recursively used to build the subtree. It results in a single-path tree model (tree model for short) where each node is a binary classification model (binary model for short). When classifying, each binary model is used to verify whether the new instance belongs to the class of that node or not. If not so, that instance is passed down to its child node and the process keep going on until the class label of the instance is decided. They applied Chi-Square as filter criterion and SVM-RBF (support vector machine with radial basis kernel [2, 3]) as target classifier. When applied for classifying five subtypes of breast cancer, their approach produces about 95% accuracy. About five features are required for each node, and totally, 18 features for the whole tree.

The authors in [9] propose the following hybrid approach. Two subsets of features are first filtered by F-Score and Information Gain, respectively. Then, the intersection set (say S_1) and the exclusive-OR set (say S_2, resulted from XOR operator) are computed from those two feature sets. At wrapper stage, Greedy Backward Elimination is conducted on S_1 while Greedy Forward Selection is executed on S_2 to find the most potential candidates. This heuristic might come from the fact that the features in S_1 can contain more information about the class than those in S_2 and the subsets of the bigger size in S_1 can be more promising than the smaller ones. In an experiment on a disorder protein data, they selected 355 out of 420 features, but no improvement in accuracy. On a lung cancer, they obtained a subset of 70 out of 7129 features which returned 100% accuracy.

In [10], the authors applied a hybrid feature selection approach, named Improved F-score and Sequential Forward Floating Search, for classification on six types of erythemato-squamous diseases. A modification of F-score was first used to filter the feature space. Then, Sequential Forward Floating Search and SVM were combined during the wrapper stage. Grid search was conducted as well to find the optimal parameters for SVM. The erythemato-squamous disease dataset, from UCI machine learning database, contains 358 samples with 34 features. Depend on the size of testing set partitioned from the data, the accuracies range from 93% to 100%.

Recently, there are new algorithms for feature selection basing on particle swarm optimization approach, such as [12, 13]. However, the comparison between these methods and PAFS are out of the scope of this study.

Study on machine learning techniques has stimulated the development of the public software for machine learning and statistic community. Among which, Weka is one of the most well-known (http://www.cs.waikato.ac.nz/ml/weka/). It has many tools for grid searching, filter and wrapper feature selection, and classification. Thus, many real life classification tasks can be solved easily via its graphical user interface. However, to the best of our knowledge, no built-in tool in Weka can combine the feature space and classifier parameter space as one. (At the moment of this study, the newest Weka version for developer is 3.9) Thus, sometimes, it hard for this software to help us find the optimal classification as expected.

III. MATERIALS AND METHODS

In this section, we introduce the TCM algorithm to deal with multi-class classification problem. It extends the tree-based scheme introduced in [1] to allow using different classifier for different binary models (node) in the same tree model. Each binary model consists of a class label, a classifier, an optimal subset of features and an optimal parameter set on which the given classifier can produce the optimal performance. PAFS, our proposed algorithm is to find such optimal binary model for a given binary problem and a classifier. The ability to find highly optimal model relies on exploring the combination space of features and parameters, instead of searching within each individual space consecutively. However, the main difference with the other mentioned works is the way PAFS traverses the search space. For datasets of low/high dimensionality, users can adjust the running parameters so that its work more exhaustive/greedy, while the search still orients to the likely targets. The idea of PAFS is adopted from Apriori algorithm [14] which is used for frequent itemset mining.

Let us define some notations before describing our proposed algorithms.

Definition 1. *Given a feature set F and a set of class labels L, a relation or dataset D on F x L, a class label i in L, and A, a subset of F. Let*

- D^{*i} *denote the new dataset obtained from D by replacing the class label of all samples to a new label, except those of class i;*
- D^{-i} *denote the sub-dataset obtained by removing all samples of class i from D;*
- $D_{|A}$ *denote the new dataset resulting from D by projecting all samples on the feature set A.*

Thus, D^{*i} is the binary problem derived from D. It is similar to a subproblem of the approach in [10], where the sample of class i are the positive ones and the others are the negative ones.

Definition 2. *Given a two-class problem D^{*i}, a set of parameters P and a real number Q. Let M = <i, C, A, P, Q> denote a binary model such that when using classifier C to train and test with parameters P on $D^{*i}_{|A}$ the average performance will be Q.*

Here, Q refers to a static measure such as accuracy, recall, area under roc curve.

Definition 3. *A s-candidate is a set of s features.*

A. Tree-based Classification Model for Multi-class Problem

Actually, PAFS can deal with multi-class problems directly, provided that the classifier used can handle multi-class problems, such as K-NN, Decision Tree, Naïve Bayes. However, for its application generality, PAFS should better be applied to two-class problems, as some classifier is originally designed to solve binary problems only, like SVM. Moreover, transforming a multi-class problem into many binary problems, and then finding the tree model to solve them seems to produce a better result than solving the original problem directly. The reason is that we can apply different feature subsets and parameters for different binary problems; instead of using the same setting to classifying all classes. Therefore, we extend the scheme for finding a tree-base model for classifying on multi-class problems in [1], and name this algorithm TCM (Fig. 1). One can see that the main step of TCM can be solved in parallel.

At step 2, for each class label i in L, we find an optimal model to discriminate instances of class i against the others. Then, the optimal model yielding the highest quality will be chosen as the binary model of the current node (step 3). After that, dataset D is shrunk by removing all samples of the

correspondingly chosen class (step 4). The class label i is removed from L as well (step 5). Then, the process continues to build the model for the subtree until only one class is left. If the original data consists of $|L|$ classes then the tree model will have $|L|$ nodes. The last node contains only the remaining class label since no more classification is needed. This is a best-first search heuristic to avoid considering too many models; otherwise, trying all permutations of the binary models would be very time-consuming.

Input:

- *F, L, D: as in Definition 1.*
- *CS: a set of classifiers to try.*

Output:

- *TM: the optimal tree model for classification on D.*

Method:

1. *Initialized TM as an empty tree.*
2. *For each class value i in L, and for each classifier C in CS, find the optimal binary model <i, C, A_i, P_i, Q_i> with respect to F, L, and D^{*i}.*
3. *Choose the model with the maximal value of Q_i to construct the root node of TM.*
4. *Update dataset $D = D^{-i}$.*
5. *Update the class label $L = L\backslash\{i\}$.*
6. *While $|L| > 1$, go back to step 2 to build the subtree of the current node.*

Fig. 1. TCM algorithm for constructing the optimal tree-based model for multi-class classification problem. Steps 2 can be implemented in parallel/distributed by assigning a binary problem to a thread/processor/computer to solve.

B. Parallel Apriori-like Feature Selection algorithm

In the one-versus-all scheme in TCM, the main task is to find the optimal binary model <*i, C, A_i, P_i, Q_i*> to predict where a new instance belongs to class i or not. Indeed, it is to find the optimal subset of features A_i, and the optimal parameters P_i for classifying on the binary classification problem D^{*i} by the given classifier C. It can be done by PAFS algorithm which is described in Fig. 2.

The idea of PAFS is to gradually generate the candidates feature subsets in a level-wise manner and select only the high-quality candidates to generate candidates for the next step. Each candidate is evaluated by the given classifier, with the corresponding optimal parameters produced by a grid search step. The process keeps continuing until the candidates' size excesses a threshold S or there is no more candidate to try. Additional stop condition can be integrated as well, e.g., the quality keeps decreasing over certain iterations.

At the initializing step, the original features are filtered by criterion, such as Mutual Information, Gain Ratio to reduce the search space significantly. Each selected feature forms a *1*-candidates, which is a highly promising one. If this filter stage is skipped, then the *1*-candidates will be selected by the classifier C.

At each following iteration, all of the candidates are of the same size (say s), and they will be tested to remove the low-quality ones. In more details, for each such s-candidate A and each parameter setting, the classifier C is trained and tested on the projected data $D_{|A}$, under 10-fold cross-validation scheme. To accelerate the process, PAFS can be implemented in parallel, where each candidate can be processed by a thread (steps 2.a.i); but a thread can handle many candidates. The optimal parameter P and its corresponding quality Q is returned (steps 2.a.ii). After processing A, the to-be-return optimal binary model is updated (step 2.a.iii). After trying all s-candidates, we compute a new minimum quality threshold, *MinQ,* (step 2.b) which is used as a threshold to remove the low-quality candidates. Here, we would like the remaining subsets to have at least 80% of the optimal one (step 2.c). By this way, the new candidates generated for the next iteration can be more promising than the current ones, hopefully. If the number of tested remaining subsets is still high, we keep only the top-N (step 2.d).

Input:

- *D: a dataset of two classes.*
- *C: a classifier.*
- *N: the maximum number of candidates to maintain after each iteration.*
- *S: the maximum size allowed for a feature subset.*
- *G: the grid of parameters to try.*
- *X: the number of features to filter.*

Output:

M: an optimal model for classification on D.

Method:

1. *Step 1. Apply a filter criterion to select top X features from the set of all features. They are the 1-candidates.*
2. *Step s (initially, s =1). Execute the following steps until s > S or there is no more candidate to try.*
   a. *For each subset A among the current s-candidates*
      i. *Use a thread/processor/computer to run 10-fold cross validation for classifier C on the projected data $D_{|A}$, for each parameter setting in G.*
      ii. *Obtain the optimal parameter P and the best quality Q .*
      iii. *If $Q > Q_M$, the quality of M, then M = <i, A, C, P, Q>.*
   b. *MinQ = 0.8*Q_M.*
   c. *Remove every s-candidate having quality smaller than MinQ.*
   d. *Keep only the top N s-candidates according to their quality.*
   e. *Generate new (s+1)-candidates from a pair of s–candidates if they share s–1 common features.*

Fig. 2. PAFS algorithm for searching the optimal classification model for two-class classification problem.

A new (*s+1*)-candidate is generated for the next step by joining a pair of s-candidates if they share *s–1* common

features. (Some new candidates can be replicated and should be removed) This heuristic not only prunes the search space but also help find the optimal subset quicker. Intuitively, if we consider the members of a *s*-candidate maintained after step 2.d as "well cooperating" in classification, then the members of the new (*s+1*)-candidate will likely "cooperate well". Because the new feature coming from one s-candidate already "cooperates well" with *s*-1 common members, it likely "cooperates well" with the left feature. As a consequence, the quality of the new candidate will tend to increase than decrease. (If we adjust *MinO* to just well under the value of O_M, e.g., 95% of O_M, then the candidates of the next generation tend to produce better quality or not much worse quality than the current optimal value.) Thus, each currently remaining candidate acts as a direction to the tops of its local hills. The more candidates are kept the more local hills can be reached, so the higher chance we can find the global optimum; but, of course, there will be more computational cost.

Note that, both *N* and *MinO* are to adjust the greedy/exhaustive behavior of PAFS, but with different effects. For example, we can increase *N* and decrease *MinO* (e.g., *MinO* = 40% of the current optimal quality) to obligate PAFS to generate and test more candidates. That is the case when running time is not a big concern (the dataset is small or simple) and the classification performance is of priority. Note that, if we set *MinO* to a very small value (e.i., MinQ = 0), then *N* top *s*-candidates will be selected. However, in such case, the selected candidates might be of too low quality. Then, they might not generate the promising candidates for the next step, and we might waste time considering them. In the reverse case (when the dataset is large, and we use the classifier of high complexity like SVM), we can decrease *N* and increase *MinO* so that, PAFS will generate and test fewer candidates to save time.

The complexity of PAFS. One can see that step 1 (filter stage) is executed only once, and it is very quick as compared to the whole process of PAFS. For each other step (step *s*), only at most $N \times (N-1)/2$ candidates are generated; each candidate is tested with every parameter setting in the grid *G*. As step *s* is repeated *S* time, the number of candidates to try can be $S \times N \times (N-1)/2$ in the worst case. Thus, the complexity of PAFS can be estimated to $O(N^2 \times S \times |G|)$, where |*G*| is the dimension of grid *G*.

Thus, *N, S,* and *G* are the primary factors to control the complexity of PAFS. Increasing their sizes, PAFS will become close to an exhaustive search; while decreasing them, PAFS will behave like a greedy one.

IV. Experimental Results and Discussions

A. Experimental datasets

We tested our approach on a gene expression data of breast cancer of five subtypes (Dataset1), which is not a public one, and on an erythemato-squamous disease dataset (Dataset2), downloaded from UCI machine learning database. They were used [1] and [10], respectively.

Dataset1 has 13582 features and 158 samples. Each sample corresponds to a patient profile, where each feature records the expression of a gene and the class indicates the subtype of cancer that the patient has. In summary, it contains 39 Basal samples, 22 Her2 samples, 53 LumA samples, 31 LumB , and 13 normal samples. Fig. 3 presents the tree-based model learned on this data by the approach in [1]. According to this model, Basal is the easiest to identify with 99.36% accuracy, followed by Normal and Her2. The classification between LumA and LumB is said to be the hardest one, with 88.1% accuracy.

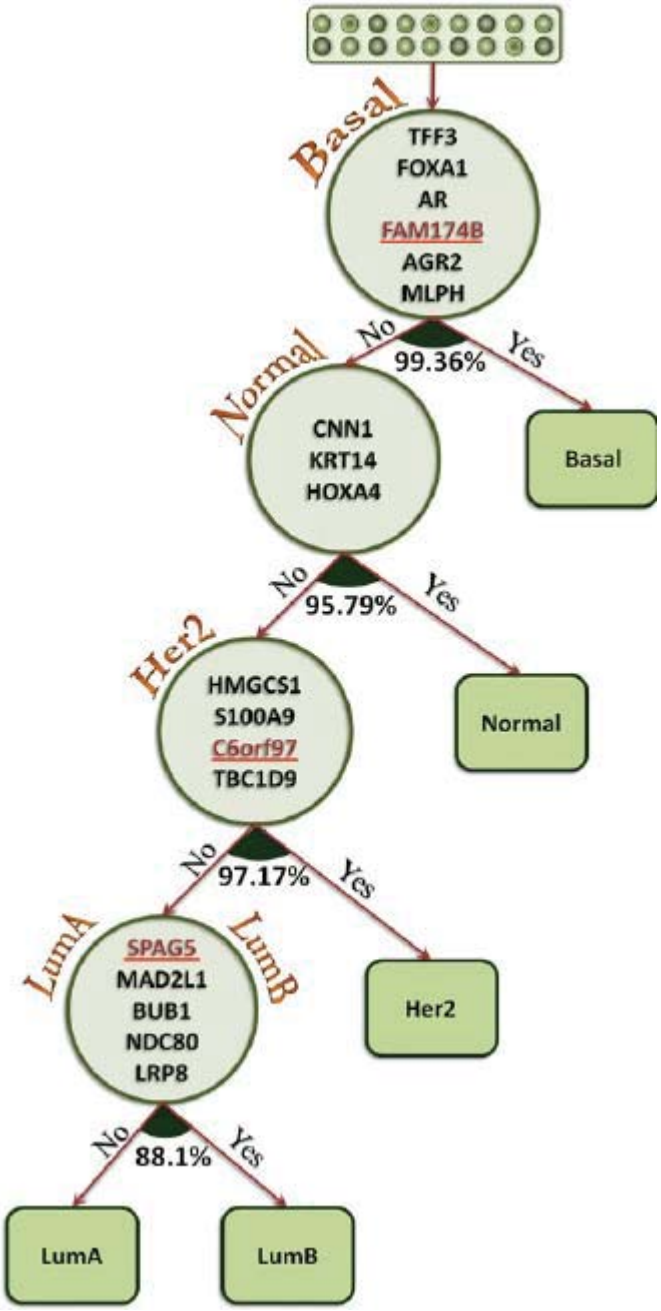

Fig. 3. Tree-based classification model resulted from [1] for breast cancer dataset of five subtypes.

Dataset2 is a dermatology disease data. It has 34 features and 366 samples. There are 12 clinical and 24 histopathological attributes. 32 out of them take integer the values 0, 1, 2, 3 indicating the degree; one is binary and another is linear. The diseases are psoriasis (1, 112), seboreic dermatitis (2, 61), lichen planus (3, 72), pityriasis rosea (4, 49), cronic dermatitis (5, 52), and pityriasis rubra pilaris (6, 20). The numbers in parentheses are class code and number of instances, respectively.

In [10], their average classification accuracy on five different partitions of Dataset2 is about 97% and their model uses about 12 attributes. The best one, which can be considered as 10-fold cross-validation, is 100% accuracy with 14 attributes.

One can realize that when transforming these datasets into binary problems in one-against-all manners, we have to deal with many unbalanced datasets.

B. Experimental notes

We have used Weka's libraries and LibSVM packages [15] to develop our own Java program, with multi-thread programming. Gain Ratio and Information Gain is used as the filter criterion options; K-NN and SVM-RBF (SVM for short) as classifiers. We refer readers to references [2, 3, 4, 5, 6] for more details about these criteria and classifiers.

In grid search, for K-NN, it is recommended to find the optimal value of K in the range 1 to $\min(15, \sqrt{N})$, where N is the number of samples. Meanwhile, for SVM, the gamma and the cost parameters are taken from the predefined arrays, such as {0, 0.05, 0.1, 0.25, 0.5, 0.75, 1, 1.5} and {1, 8, 16, 64, 128}, respectively. Since trying all setting of parameters would be time-consuming, we have running options to traverse only a few first values of those arrays. For example, we can run PAFS with only the first value of the parameter predefined ranges (i.e., K = 1 for K-NN and *cost* = 0, *gamma* = 1 for SVM). In this case, we run PAFS without grid search, and it corresponds to running those classifiers with default parameters in Weka.

The running option setting for TCM is provided in form of *<classifiers used, criterion for filtering stage, number of features to filter, maximum size of a feature subset (i.e., parameter S in PAFS), maximum number of subsets to maintain after testing (i.e., parameter N in PAFS), with grid search or not>* as in the captions of Table I - Table V. We evaluate the model quality in term of Matthews correlation coefficient (MCC), one of the best scores to deal with imbalanced data [16]. MCC is in the range [-1, 1], and the higher the value of MCC the better the quality of classification. We recorded also the model accuracies to comparison with the result in [1].

PAFS is implemented in multithreading fashion. For step 2.a, if we use T threads to process Z s-candidates generated at each step, then each thread will process about Z/T candidates. For each feature subset A and parameter setting P, the classification performance (accuracy/MCC) is evaluated in 10-cross validation scheme. That is, the projected dataset $D_{|A}$ is first stratified and partitioned into 10 subsets. Then, at each time, a subset is used for testing and the combination of the 9 others is used for training. This process is repeated 10 times.

C. Results and discussions

In our study, each tree model for multi-class classification problem is presented as a table, where each row is a binary model. The order of the rows corresponds to the order of nodes in the tree, and the last node is omitted. For simplicity, we consider the overall accuracy of each tree model is the average accuracy of its nodes except the omitted node. For example, for the breast cancer dataset, the overall accuracy is the average of 4 nodes, since we have totally 5 subtypes. For each binary model corresponding to a node, we tested again by Weka software under the same setting (data, classifier, selected features, and parameters, etc) and got exactly the same result. Thus, there is no logical mistake in our implementation.

On Dataset1, under many running options, we obtained the very good results: the overall accuracies are greater than 99% and the overall MCCs are about 0.99. Most of them return the following classification order from easiest to hardest to classify: *Basal against all → Her2 against Normal, LumA and LumB → Normal against LumA and LumB → LumA against LumB*. This is shown in Tables I, II, III, IV; in which class *LumB* is omitted because it is the last one to identify.

Table I gives information about the result (tree model) when using SVM as the classifier, Information Gain to filter top 100 features at filter stage, with grid search for SVM parameters. The maximum size of a feature subset to try is 5. Only at most top 40 subsets are maintained after each iteration to generate the candidates for the next iteration. Each row in the table is a binary model. For example, the first row in Table 1 means that when classifying for a new patient, the first binary model examines on features/genes {TFF3, AGR2} and uses SVM with gamma = 0, cost = 1 to verify if the patient has Basal cancer or not. This test can yield 100% of accuracy. If the patient is predicted not to have Basal cancer, then the binary model of the second row is used next, and so on. At the last row, if that patient is not predicted as LumA cancer then he/she is predicted as LumB. The overall accuracy is 99.49%.

Table II presents a perfect model with 100% accuracy overall, using SVM. Table III presents result when using the same running options as the case of Table I, except that {K-NN, SVM} is the classifier space. Here, SVM is chosen if SVM produces the same quality as K-NN. Thus, the model in this table in almost similar to that of Table I, but the second row is replaced by a model of K-NN which has better quality than that of SVM in the second row of Table I. This is a tree model using different classifiers for classification.

Table IV presents a result when running with almost the same options as in Table I but without grid search, i.e., gamma is set to 0 and cost is set to 1. As it shows here, without grid search, we can still obtain higher accuracy than the result in [1]. This supports for the effectiveness of the way that AFS explores the feature space. However, when comparing to the results in Table I, its accuracy and MCC is lower. This implies that when integrating the parameter search, we can improve the classification further. This means that searching for the optimal

model in the combination space of features and parameters is a reasonable approach.

TABLE I. TREE-BASED MODEL LEARNED FROM DATASET1 BY TCM USING RUNNING OPTIONS <SVM, INFORMATION GAIN, 100, 5, 40, WITH GRID SEARCH>, 98.70% ACCURACY.

Class	Selected features	Parameters	Accuracy	MCC
Basal	TFF3, AGR2	0, 1	100.0%	1
Her2	HMGCS1, SLC39A6, TARS, YBX1	0.05, 64	99.16%	0.973
Normal	CX3CL1, ARAP3	0, 1	100.0%	1
LumA	MAD2L1, SRSF5, CBX8, KATNB1	0.75, 8	98.81%	0.975

TABLE II. TREE-BASED MODEL LEARNED FROM DATASET1 BY TCM USING RUNNING OPTIONS <SVM, INFORMATION GAIN, 150, 5, 40, WITH GRID SEARCH>, 100% ACCURACY.

Class	Selected features	Parameters	Accuracy	MCC
Basal	TFF3, AGR2	0, 1	100.0%	1
Her2	TARS, YBX1, MDP1, ATP1A1OS	1, 1	100.0%	1
Normal	CX3CL1, ARAP3	0, 1	100.0%	1
LumA	SPAG5, NDC80, LRP8, MRPS23, NEK2, CACYBP, RCL1, LRIG1, HMMR	0, 16	100.0%	1

TABLE III. TREE-BASED MODEL LEARNED FROM DATASET1 BY TCM USING RUNNING OPTIONS <K-NN/SVM, INFORMATION GAIN, 100, 5, 40, WITH GRID SEARCH>, 99.70% ACCURACY.

Class	Classifier	Selected features	Parameters	Accuracy	MCC
Basal	SVM	TFF3, AGR2	0, 1	100.0%	1
Her2	K-NN	C2orf54, FAM134B, DROSHA	2	100.0%	1
Normal	SVM	CX3CL1, ARAP3	0, 1	100.0%	1
LumA	SVM	MAD2L1, SRSF5, CBX8, KATNB1	0.75, 8	98.81%	0.975

TABLE IV. TREE-BASED MODEL LEARNED FROM DATASET1 BY TCM USING RUNNING OPTIONS <SVM, INFORMATION GAIN, 100, 5, 40, WITHOUT GRID SEARCH>, 98.32% ACCURACY.

Class	Selected features	Parameters	Accuracy	MCC
Basal	TFF3, AGR2	0, 1	100.0%	1
Her2	S100A9, BZRAP1, THSD4, CEP55	0, 1	99.16%	0.972
Normal	CX3CL1, ARAP3	0, 1	100.0%	1
LumA	NUSAP1, KIF4A, CACYBP, UTP18	0, 1	94.05%	0.875

Compare to the result in [1] (Fig. 3), all of our four tree models have higher accuracy, both in average and in each corresponding binary model. Especially, the accuracy of the binary model for classifying LumA, and LumB (the hardest case) in Table II, reaches 100%, about 12% higher than that of [1]. The order of the second and the third nodes in our model are the reversion of that in [1] since PAFS detected that Her2 cases are easier to be identified than the Normal cases. Additionally, the model in [1] they needs 18 genes in total; meanwhile, each of our tree models need fewer genes: 12, 17, 13, 12 genes, respectively. Even without grid search, as shown in Table IV, our model still obtain higher accuracy. This supports for the effectiveness of the way that AFS explores the feature space.

In comparison the classification of LumA and LumB (the last row) in Table I, Table II, Table III to that in Table IV, one can see that the integration of grid search with feature subset search has taken significant effect in improving the classification. This means that searching for the optimal model in the combination space of features and parameters is a reasonable approach.

In Table V, we present the tree model learned from Dataset2 when running with the option: <*SVM, Gain ratio, 34, 4, 40, with grid search*>. This option means that we did not filter any features at filter stage (filter 40 out of 34 features). The class labels and the features are shown as their original code from the data source. (Class number 4 is omitted because it is the last one to identify) Its overall accuracy is 99.26%, just slightly smaller than that the best case in [10] but it needs only 10 features, while that in [10] requires 14 features in the best case.

TABLE V. TREE-BASED MODEL LEARNED FROM DATASET2 BY TCM USING RUNNING OPTIONS <SVM, GAIN RATIO, 34, 4, 40, WITH GRID SEARCH>, 99.26% ACCURACY.

Class	Selected features	Parameters	Accuracy	MCC
1	20, 22	1, 8	100.0%	1
3	6, 8	0.75, 1	100.0%	1
5	1, 15	0, 1	100.0%	1
6	1, 7	0.05, 1	100.0%	1
2	5, 26	0, 1	96.3	0.925

D. A brief results of another study.

We also applied our method on 9 breast cancer datasets, of the Molecular Taxonomy of Breast Cancer International Consortium Study [17], for predicting the survival time of patient after receiving hormone and chemotherapy treatment. Most of them contain about 400 to 500 samples, with class imbalance in data. Previously, in [18], mRMR and SVM (in Weka) were used to find the most informative genes for prediction. mRMR was tried with several searching methods such as Linear Floating Forward, Linear Floating Backward, Greedy Stepwise and SVM was used for evaluating in wrapper approach. The accuracies range from 66% to 84%; however, the values of area under ROC curve (AUC) are not high in some cases. Meanwhile, our approach can return higher accuracies, AUC, and MCCs. In three cases, both accuracies and MCCs increase about 20%, while in other cases, the MCCs are doubled.

V. CONCLUSIONS AND FUTURE WORKS

The following three observations inspire the idea of PAFS. (1) Feature selection is a key factor in pruning the search space, reducing the noise, and increase the prediction quality, while maintaining the characteristics of the original data. (2) Parameter search is another technique to improve the power of classifier and grid search is one of the suitable ways to examine the parameter space. And (3) the classifier performance change from one dataset to another dataset and from one parameter

setting to another. Thus, by combing the feature space and the parameter space into one, PAFS can find the classification model that is more optimal than the approaches exploring those spaces separately. Its search behavior is controllable by parameters so it cannot be too exhaustive or too greedy. Based on PAFS and one-versus-all strategy, we presented in TCM that makes use of the power of different classifiers to deal with multi-class problems.

When applying the model for classification on a breast cancer of five subtypes dataset, and on an erythemato-squamous disease dataset, our models can yield more than 99% accuracy but require fewer features than previous studies. This supports that our approach is effective. Besides, PAFS can be applied for regression problems. It can be done by replacing the classifier with a regression method, such as K-NN, Neural Network, SVMRegression (available in Weka) and replacing the quality measurement (i.e., MCC in this paper) by a criterion suitable to that regression method (e.g., Minimum least squared error). Reference [13] is an example of applying feature selection for regression.

In the future study, we intend to conduct more experiments on different and larger datasets to see if our approach fit well with sparse data or dense one. Besides, experiments on running time and on the number of candidates generated and tested when PAFS parameter change might give more information about its behavior and effectiveness. We also plan to implement PAFS in a distributed environment so that it can be able to solve run on much larger datasets in a reasonable time. The comparison with other new methods will be conducted as well.

ACKNOWLEDGMENT

We thank reviewers for useful comment and suggestions so that we can improve the paper and figure out what could be investigated further.Acknowledgment

We thank reviewers for useful comment and suggestions so that we can improve the paper and figure out what should be investigated further.

REFERENCES

[1] Iman Rezaeian, Yifeng Li, Martin Crozier, Eran Andrechek, Alioune Ngom, Luis Rueda, and LisaPorter, "Identifying informative genes for prediction of breast cancer subtypes", Pattern Recognition in Bioinformatics, Springer Berlin Heidelberg, 2013.

[2] Sergios Theodoridis and Konstantinos Koutroumbas, "Pattern Recognition, 4th Edition", Academic Press, 2009.

[3] Dustin Boswell, "An introduction to Support Vector Machine", 2002, PDF file available at http://dustwell.com/past-work.html.

[4] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin, "A practical guide to support vector classification", 2003.

[5] Judea Pearl, "Causality: Models, Reasoning, and Inference", Cambridge University Press, 2000.

[6] Tom Mitchell, Machine Learning, McGraw Hill, 1997, pp. 52-80.

[7] Chris Ding, and Hanchuan Peng, "Minimum redundancy feature selection from microarray gene expression data", Journal of Bioinformatics and Computational Biology, Vol. 3, No. 2, 2005, pp.185-205.

[8] Hanchuan Peng, Fuhui Long, and Chris Ding, "Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy", Pattern Analysis and Machine Intelligence, IEEE Transactions, Vol. 27, No. 8, 2005, pp.1226-1238, 2005.

[9] Hui-Huang Hsu, Cheng-Wei Hsieh, and Ming-Da Lu. "Hybrid feature selection by combining filters and wrappers", Expert Systems with Applications, Elsevier, Vol. 38, Issue 7, 2011, pp.8144-8150.

[10] Juanying Xie, Weixin Xie, Chunxia Wang and Xinbo Gao, "A Novel Hybrid Feature Selection Method Based on IFSFFS and SVM for the Diagnosis of Erythemato-Squamous Diseases", JMLR: Workshop and Conference Proceedings 11, 2010, pp. 142–151.

[11] Pineda-Bautista, Carrasco-Ochoa, and Martınez-Trinidad. "General framework for class-specific feature selection", Expert Syst Appl., Vol. 38, Issue 8, 2011, pp. 10018-10024.

[12] Hoai Bach Nguyen, Bing Xue and Peter Andreae, "Mutual Information Estimation for Filter Based Feature Selection Using Particle Swarm Optimization", European Conference on the Applications of Evolutionary Computation, Springer International Publishing, 2016.

[13] Hu, Z., Bao, Y., and Xiong, T., "Comprehensive learning particle swarm optimization based memetic algorithm for model selection in short-term load forecasting using support vector regression", Applied Soft Computing, 25, 2014, pp 15-25.

[14] Rakesh Agrawal, and Ramakrishnan Srikant, "Fast algorithms for mining association rules", Proc. 20th int. conf. very large data bases, VLDB, Vol. 1215, 1994, pp. 487-499.

[15] Chih-Chung Chang and Chih-Jen Lin, "LIBSVM : a library for support vector machines", ACM Transactions on Intelligent Systems and Technology, 2:27:1--27:27, 2011, Software available at http://wwsw.csie.ntu.edu.tw/~cjlin/libsvm.

[16] Giuseppe Jurman, Samantha Riccadonna, Cesare Furlanello"A Comparison of MCC and CEN Error Measures in Multi-Class Prediction", PLoS One. 2012; 7(8): e41882, 2012, doi: 10.1371/journal.pone.0041882, PMCID: PMC3414515.

[17] Huy Q. Pham, Iman Rezaeian, Eliseos J. Mucaki, Alioune Ngom, Luis Rueda, Peter K. Rogan, "Predicting Breast Cancer Drug Response via a Level-wise Gene Selection Approach", ISCB-Latin America 2016, 2016, unpublished.

[18] Rezaeian I, Mucaki EJ, Baranova K *et al,* "Predicting Outcomes of Hormone and Chemotherapy in the Molecular Taxonomy of Breast Cancer International Consortium (METABRIC) Study by Biochemically-inspired Machine Learning", [version 1; referees: 2 approved with reservations], F1000Research 2016, 5:2124 (doi:10.12688/f1000research.9417.1).