Online Random Forests based on CorrFS and CorrBE

HASSAB ELGAWI Osman
Tokyo Institute of Technology
osman@isl.titech.ac.jp

Abstract

This paper aims to contribute to the merits of online ensemble learning for classification problems. To this end we induce random forests algorithm into online mode and estimate the importance of variables incrementally based on correlation ranking (CR). We test our method by an "incremental hill climbing" algorithm in which features are greedily added in a "forward" step (FS), and removed in a "backward" step (BE). We resort to an implementation that combine CR with FS and BE. We call this implementation CorrFS and CorrBE respectively. Evaluation based on public UCI databases demonstrates that our method can achieve comparable performance to classifiers constructed from batch training. In addition, the framework allows a fair comparison among other batch mode feature selection approaches such as Gini index, ReliefF, and gain ratio.

1. Background and Overview

Many real-life machine learning problems can be more naturally viewed as online rather than batch learning problems. With the emergence of ensemble learning, which is highly successful with batch learning, interest to on-line algorithms that are able to learn continuously for this learning approach arose as well. Ensemble learning algorithms are designed to handle large feature[1] spaces even when it is expected that only a small number of features will actually be useful. However, most of the current ensemble learning approaches work off-line or in a "pseudo-on-line" batch processing mode (i.e. collecting a set of training examples and then run the off-line ensemble algorithms). Decision tree ensemble-based methods such as *bagging* [1], *boosting* [4], *Random Forests* (RF) [2] and their invariants also generate extra information that allow to estimate the importance of each explanatory variable. e.g., based on permutation accuracy and on impurity decrease, RF measures variable importance by randomly permuting the values of the variable m for the out-of-bag[2] (OOB) cases for tree k, if variable m is important in the the classification, then the accuracy of the prediction should decrease. On the other hand, we can consider the accumulated reduction at nodes according to the criteria used at the splits, an idea from the original CART [3] formulation. Variable importance measures can be used to perform variable selection. Alternatively, there has been substantial work on engineering feature spaces which can be divided into three branches: 1) feature ranking [14], ranking feature with respect to their *relevance*, 2) subset selection [7] where we must select a subset k of d dimension that give us the most information according to some criteria and cast away the other $(d - k)$, which is considered *irrelevant* or *redundant* and 3) domain experts contain only those attributes which are expected to be relevant to the classification problem. Currently, there has been an increased interest in applying kernel functions for providing implicit feature space. While the feature selection literature is ample, most of approaches work in batch mode. However, batch learning of feature selection limits the usage for a wide variety of machine learning problem. This paper contributes to the merits of online ensemble learning for classification problems. To this end we induced random forests algorithm into on-line mode and hypothesize that generating relevant features incrementally on the basis of correlation ranking (CR) can perform comparably to the counterpart batch mode ensemble learning algorithms. We test our model by an "incremental hill climbing" algorithm in which features are greedily added in a "forward selection" step (FS), and removed in a "backward elimination" step (BE). We favor the Breiman's RF for our ensemble learning implementation since it yields improvement over bagging, fast training, proven not to overfit, and computationally effective ($O(\sqrt{M}NlogN)$, where M is the number of variables and N is the number of observations). It should be noted that the benefit of this study is inside into interplay of combining variable selection and ranking method into on-line forest. The motivation for use of this method

[1]Feature (attribute) selection, feature sub-set selection and variable selection although are distinct but often used interchangeably in literature.

[2]There is on average $I/e \approx 36.8$ of instances not taking part in construction of the tree, provides a good estimate of the generalization error (without having to do cross-validation).

specifically with ensemble methods and RF in particular is elaborated in the discussion session. Furthermore, the framework allows a fair comparison between other batch mode feature evaluation approaches; Gini index [3], ReliefF [8, 11, 12], and gain ratio [10]. These methods evaluate features based on different criteria that include impurity decrease, Euclidean distance and information entropy respectively. For a valuable discussion on several measures for estimating attribute's quality in batch learning classification problem, the reader is directed to [5, 8, 9]. The remainder of this paper is organized as follows. Section 2 contains some discussion on on-line learning RF with incremental feature selection based on correlation ranking. In Section 3 we evaluate the efficiency of our approach via extensive experiments on 12 UCI domains comparing with other state-of-art feature selection methods and batch learning algorithms. Results and brief discussion will appear in Section 4. We will conclude with a summary in Section 5.

2. On-line Incremental Learning RF

In this section we explore the possibility of our on-line incremental RF algorithm. Before proceeding with our presentation we need to establish some definitions and notations. Then briefly review the off-line random forests approach and how it is used for estimating variable importance. Finally, we present our novel on-line random forests for incremental feature selection. Informally, the goal of the on-line algorithm is to minimize the number of prediction errors. Given n instances of different training set T consisting of (x_i, y_i) pairs, where the x_i values are represented feature vectors, and the y_i represents the value to be predicted. In a classification problem $y_i, i = 1, \ldots, c$,(c is number of class values) represents one or more classes, several N bootstrap samples $(B_1, B_2, \ldots, B_n)$ are drawn from T, and unpruned decision tree is constructed by first choosing an important attribute $f(1 \leq f \leq m)$ of the elements of $\{x_1, x_2, \ldots, x_m\}$ (m is the number of variables) according to evaluation criterion. While the instances may belong to a well defined partition of feature space into classes, we do not receive direct supervision in the form of class labels. Instead, we get correlation ranking feedback. Variables are rank based on importance scores $S(x_i)$ and similarity scores. We write CR_j for the correlation rank of feature j and P_e for the probability of error estimate (*irrelevance*).

2.1. Off-line Random Forests

Random Forests (RF) is tree-based ensemble prediction technique for high dimensional classification and regression [2]. Briefly, it is an ensemble of two sources of randomness to generate base decision trees; bootstrap replication of instances for each tree and sampling a random subset of features at each node.

Definition 1: (Random Forest) For the k-th tree, a random vector θ_k is generated, independent of the past random vectors $\theta_k, \ldots, \theta_{k-1}$, but with the same distribution, and a tree is grown using the training set T and θ_k , resulting in a classifier $h(x, \theta_k)$ where the θ_k are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input random vector x. The forest is an ensemble of tree-generated classifiers $h(x, \theta_k)$, $k = 1, \ldots, n$.

For the n-th sample in the data, RF computes the margin *(mr)* at the end of a run, which is the proportion of votes for its true class minus the maximum of the proportion of votes for all other classes.

$$mr(X,Y) = \\ P_\theta(h(x,\theta) = Y) - \max_{j \neq Y} P_\theta(h(x,\theta) = j) \tag{1}$$

According to Breiman the error rate of a forest depends on the correlation between any two trees and the strength of each tree in the forest. One can arrive at OOB prediction as follows: for a case in the original data, predict the outcome by plurality vote involving only those trees that did not contain in the case in their corresponding bootstrap sample. By contrasting these OOB predictions with the training set outcomes, one can arrive at an estimate of the prediction error rate, which is referred to as the OOB error rate.

2.2. Off-line RF for Variable importance

As part of their construction RF predictors naturally lead to variable importance measures. However their variable importance measures show a bias towards correlated variables. The measure of importance of the m-th variable is the average lowering of the margin across all cases when the m-th variable is randomly permuted. The random selection of variables makes individual trees rather weak. Several methods has been made for strengthening individual trees by using feature estimation measures such as ReliefF, gain ration, etc. e.g., ReliefF proved to be effective in problems which possibly involve much feature interactions. Another attempts stem from voting mechanism. In spite of apparent success of RF methodology, thus far, there is no potential work has been done in extending random forests for on-line learning, since it is difficult to design an incremental solution for importance variables. In the following sections we elaborated our idea with some results. The outcomes of this paper though simple, may be of great stimuli.

2.3. Correlation-based Feature Ranking

Two general function evaluation to guide variable sub selection exist: *filters* and *wrappers* [7]. Both methods distinct between variable selection and learning process. A hybrid model have been proposed to combine the advantages of both algorithms [15, 13]. In another implementation variable selection can be viewed as integral process of learning [14, 16], search through the space and generating potential variables based on their *relevance* to classification problem. Recently many variable selection algorithms include variable ranking as a principal selection mechanism because of its simplicity and scalability. However, unlike filter methods, the ranking criteria we proposed is based by taking into account interaction between features using *Correlation Ranking (CR)*. On each step and incrementally we estimate the importance of variable and for these criteria we have tested different search-space methods; sequential feature selection (FS) and recursive background feature elimination (BE).

$$CR_j = \frac{|\left(x_j - \mu_j\right)^T \left(y - \mu_y\right)|}{|x_j||y|}, j = 1, 2, \dots, D_{Feat} \quad (2)$$

where CR_j is rank of feature j, x_j is feature vector j, y is class label vector, μ_j and μ_y are expectation values of feature j and class vector y respectively, and D_{Feat} is dimensionality of feature space.
Then, we boil down the problem of variable ranking to find a suitable measure of correlation between variables. We first define the importance score $S(x_i)$ from bootstrap samples by permutation of $F : \grave{F} = \{x_{i1}, \dots, x_{ij}, \dots, x_{in}\}$ with $S(x_{ij}) \geq S(x_{ij+1})$; $J = 1, \dots, n-1$. Then we empoly an efficient algorithm to maximize the total importance scores and minimize the total similarity scores of a set of features.

$$\begin{aligned} &\max \sum_i \omega_i x_i \\ &\min \sum_i \sum_{j \neq 1} e_{i,j} x_i x_j \\ s.t. \quad & x_i \in \{0,1\} \quad i = 1, \dots, m \\ &\sum_i x_i = t \end{aligned} \quad (3)$$

Here t denotes the number of selected features, $x_i = 1$ (or 0) indicates that feature v_i is selected (or not), ω_i denotes the importance score of feature v_i, and $e_{i,j}$ denotes the similarity between feature v_i and feature v_j. We let $e_{i,j} = e_{j,i}$. Another notion of similarity score is irrelevance. Perfectly correlated variables are truly irrelevant in the sense that no additional information is gained by adding them. The central hypothesis is that relevance features sets contain features that are highly correlated with class, yet uncorrelated with each other. The first variable selected is the variable with the smallest probability of error (POE) (P_e). The next variable selected is the variable that produces the minimum weighted sum of (P_e), and average correlation rank (ACR), and so on. ACR is the mean of the correlation ranking scores of the candidate variable with the variables previously selected at that point. This method can rank the entire variable. Usually we would continue the process until no improvement on the evaluation function is found. However, we have noted that, at certain points, the change of function score is very small. Because of that, in our implementation a required number of features (M) is used as the stopping criterion.

Correlation Ranking (CR):
Start:
```
POE + ACR $(N, M, \omega_1, \omega_2)$
[1] $F = \phi$
[2] Find the feature with minimum $P_e$ and
append it to $F$
[3] For $i = 1$ to $M - 1$
      Find the next feature with minimum
      $\omega_1(p_e) + \omega_2(ACR)$
      Append it to $F$
[4] Return $F$
```
End:

Estimate for m samples will be as follow:

$$CR_j = \frac{\sum_{k=1}^{m}(x_{k,i} - \bar{f}_i)(y_k - \bar{y})}{\sqrt{\sum_{k=1}^{m}(f_{k,i} - \bar{f}_i)^2 \sum_{k=1}^{m}(y_k - \bar{y})^2}} \quad (4)$$

2.4. Incremental Variable Selection

Our incremental feature selection implementation is performing the same sort of *incremental hill-climbing* search for generating a concept hierarchy known as *sequential selection* which may be either *forward (FS)* or *backward (BE)*. We resort to an implementation that combine CR with FS and BE. We call this implementation CorrFS and CorrBE respectively. It should be noted that features arrive in stages after being ranked using our correlation ranking method mentioned above. CorrFS, start with no variables $F_0 = \phi$ and then during each run m adds a new set $\grave{f}$ of features. As can be seen in Eq.(5), the set of all features at stage m is denoted by F_m, where F_m is the union of features that have just arrived with the set of features that was selected at stage $m-1$. At each step and greedily adding the one that enhances the evaluation and decreases the error most, until any further addition does not significantly decreases the error.

$$F_m = F_{m-1} \cup \{\grave{f}\}, \quad (5)$$

where

$$\grave{f} = arg \max_{F_{m-1} \cap \{\grave{f}\} = \phi} Q\left(F_{m-1} \cup \{\grave{f}\}\right) \quad (6)$$

Correlation Foward Feature Selection (CorrFS):
Start:
```
[3.1a] Calculate variable ranking $F(0)$
[3.2a] LET feature $f(0)$= $\phi$; $error(0)$=$+\infty$
[3.3a] FOR $m = 1$ TO the number of features
         in random subset
[3.4a] LET $f(m) = f(m-1) \cup$ the m-th best
         feature in $F(0)$
[3.5a] Perform a training with $f(m)$ to
```

```
          obtain error rate $error(m)$
[3.6a] IF $error(m) > error(m-1)$ THEN
[3.7a] terminate feature select and
          RETURN $f(m-1)$
[3.8a] NEXT $m$
```
End:

On the other hand CorrBE, start with all variables $F_0 = F$ and removes $f^{\cdot}$ from F_0 the next worse ranked feature and trains an induction algorithm with the feature set $F_0 - \{f^{\cdot}\}$. The feature whose removal minimizes the error in the resulting classifier is removed, and $F_1 = F_0 - \{f_{max}\}$. In the next round, for each of the remaining features $f^{\cdot} \in F_1$, the algorithm tests the feature subset $F_1 - \{f^{\cdot}\}$ and removes the feature f_{max} that minimizes the error of the resulting classifier to produce $F_2 = F_1 - \{f_{max}\}$. This process repeats and at each step removes the one that decreases the error most until a local minimum of classifier error has been reached (any further removal increases the error significantly) or some other stopping condition is met. In practice, removing a feature may have only a very small negative impact on performance, so backward elimination become a trade-off between marginally lower performance and a smaller set of features. This implies a need for some sort of regularization. following [7], we added a penalty $c = 0.001$ per feature - or equivalently, the zero-norm of the weight vector - to force the algorithm to favor smaller subsets.

$$F_m = F_{m-1} \backslash \{f^{\cdot}\}, \tag{7}$$

where

$$f^{\cdot} = arg \max_{F_{m-1}\backslash\{f^{\cdot}\} \neq \phi} Q\left(F_{m-1} \cup \{f^{\cdot}\}\right) \tag{8}$$

Correlation Backward Feature Elimination (CorrBE):
Start:
```
[3.1b] Calculate variable ranking $F(0)$
[3.2b] LET $error(0) =$ average error rate;
          LET feature subset $f(0) =$ all
[3.3b] FOR $m = 1$ TO the number of all
          features
[3.4b]    REPEAT
[3.5b] Let $f(m) = f(m-1)$- the next worst
          feature in $F(m-1)$,
[3.6b]    Perform a training cycle with
          $f(m)$, calculate ranking $F$
[3.7b] UNTIL $\text{error}(m) \leq \text{error}(m-1)$OR no more
          next worst feature
[3.8b] IF no more next worst feature
          THEN stop feature selection
          RETURN $f(m-1)$
[3.9b] LET $F(m) = F$
[3.10b] NEXT $f$
```
End:

2.5. On-line RF with Incremental Feature Selection

Inspired by the success of batch learning RF, we devise an on-line scheme for the integration of incremental selection of variable importance. The structure of on-line forest is shown in Figure 1. Based on variables ranking we develop a new, conditional permutation scheme for the computation of variable importance measure. The resulting incremental variable importance is show to reflect the true impact of each predictor variable more reliably than the original marginal approach. According to features ranking results, different yet random feature subsets are used as new feature spaces for learning a diverse base-classifiers. The diversity stem from fully growing (unpruned) independent tree- base learners. Individual trees in RF are incrementally generated by specifically selected subsamples from the new feature spaces. In constrast to off-line random forests, where the root node always represents the class in on-line mode, for each training sample, the tree adapts the decision at each intermediate node (nonterminal) from the response of the leaf nodes, which characterized by a vector (w_i, θ_i) with $\|w_i\| = 1$. Root node numbered as 1, the activation of two child nodes $2i$ and $2i+1$ of node i is given as

$$u_{2i} = u_i.f(w_i^{'}x + \theta_i) \tag{9}$$

$$u_{2i+1} = u_i.f(-w_i^{'}x + \theta_i) \tag{10}$$

where x is the input pattern, u_i represents the activation of node i, and $f(.)$ is chosen as a sigmoidal function. Consider a sigmoidal activation function $f(.)$, the sum of the activation of all leaf node is always unity provided that the root node has unit activation. The forest consist of fully grown trees of a certain depth l. The general performance of the on-line forests depends on the depth of the tree. It is not clear how to select the depth of the on-line forests. One alternative is to create a growing on-line forests where we first start with an on-line forest of depth one. Once it converges to a local optimum, we increase the depth. Thus, we create our on-line forest by iteratively increasing its depth. However, we found that the nubmber of trees one needs for good performance eventually tails off as new data vectors are considered. Since after a certian depth, the performance of on-line forest does not vary to a great extent, the user may choose K (the number of trees in forest) to be some fixed value or may allow it to grow up to the maximum possible which is at most $|T|/N_k$, where N_k the user-chosen number of the size of each decision tree. When classifying a new instance, we first estimate the average margin of the trees on the instances most similar to the new instance and then, after discarding the trees with negative margin, weight the tree's votes with the margin. Then, test sets (Table.1) T_{test} that needs to be classified is put down each of the tree in the forest for classification. On-line RF was trained on each of the test set excatly once. Each tree gives a vote that indicates the tree's decision about the class of the data set. The forest chooses the class with the most votes for the object. We measure the error of h on the T_{test} as the proportion of test cases that h misclassifies:

$$\frac{1}{|T_{test}|} \sum_{(x,y) \in T_{test}} I(h(x) \neq y) \tag{11}$$

Where h is the approximation hypothesis, T_{test} is the test set and $I(v)$ is the indicator function - it return 1 if v is true and 0 otherwise.

When a new sample arrives the tree-base learners are updated. For updating the tree-base classifiers, any on-line learning algorithm can be used, however, we update with respect to the importance of the current sample and returns new hypothesis updated with

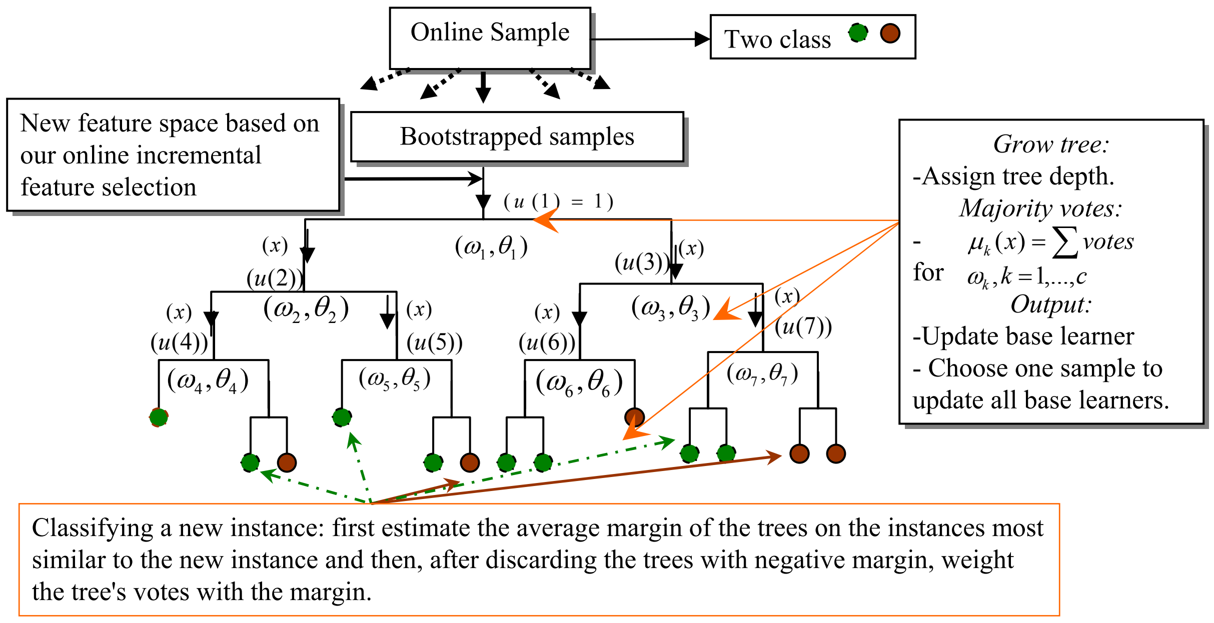
Online Sample
Two class
New feature space based on our online incremental feature selection
Bootstrapped samples
Grow tree:
-Assign tree depth.
Majority votes:
- $\mu_k(x) = \sum votes$
for $\omega_k, k = 1,...,c$
Output:
-Update base learner
- Choose one sample to update all base learners.
$(u(1) = 1)$
(ω_1, θ_1)
(ω_2, θ_2)
(ω_3, θ_3)
(ω_4, θ_4)
(ω_5, θ_5)
(ω_6, θ_6)
(ω_7, θ_7)
Classifying a new instance: first estimate the average margin of the trees on the instances most similar to the new instance and then, after discarding the trees with negative margin, weight the tree's votes with the margin.

Figure 1. Structure of on-line forest. Each intermediate node accepts the input pattern vector x as input,and it activates its two child nodes differentially depending on the embedded sigmoidal function defined by its parameters (ω, θ). The root node activation $u(1)$ is always unity.

new sample. One sample is used to update all base classifiers and the corresponding voting weight.

3. Empirical Evaluations

3.1. Datasets

We experimented and conducted evaluation using 12 publicly available data sets, all from UCI repository Machine Learning Database [6], and they are summarized in table 1 which contain information on the size of instance (real datasets must have more than 300 instances), number of classes, number of numeric features (num), number of nominal features (nom), number of all features together (All), and percentage of missing value. Missing data is handled by treating each missing value of attribute f as if it were equal to the median $\hat{f}$ of f of the training set. For the letter data set we use a randomly selected subset of 5000 instances from the whole set of 20000 instances. All our experiments shown below run on standard PC (Intel Pentium 1.6 GHZ with 512 MB RAM).

Table 1. Summary of the characteristics of the datasets. The % missing column shows what percentage of the data set's entries have missing values

			Features			miss
Dataset	Data size	Classes	Num	Nom	All	%
Pima	768	2	8	0	8	0
glass	214	7	9	0	9	0
iris	150	3	4	0	4	0
soybean	683	19	0	35	35	9.78
segmen	2310	7	19	0	19	0
letter	20000	26	16	0	16	0
satimage	6435	6	36	0	36	0
adult	5908	2	6	8	14	86
vehicle	846	4	18	0	18	0
vowel	990	11	10	1	11	0
sonar	208	2	60	0	60	0
vote	435	2	0	16	16	5.63

3.2. Experimental setting

We conducted two set of comparisons. In one set of experiments, as a baseline for comparisons, the standard random forests based on Gini index as splitting criterion was run on the datasets. Then against the state-of-the-art feature selection methods; ReliefF and gain ratio. Since these feature selection methods used for comparison here are well known in the machine learning community and proved popular in practice, we attempted to investigate whether our methods' (CorrFS and CorrBE) prediction performance compares favourably or not. Since the classification accuracy (Acc) is the primary evaluation criterion, when comparing a pair of methods, we present the prediction accuracy (Acc) of a learning algorithm on each datasets. In anther set of experiments, in order to verify how the prediction accuracy induced by on-line RF could improve by employing incremental feature selection based on correlation ranking, we ran four batch learning algorithms; standard Random Forests (RF), AdaBoost, Support vector machine (SVM), and K-means Nearest Neighbor (KNN) on all our datasets. The two variants of our on-line RF algorithms, referred

Table 2. Accuracy of RF using different feature selection methods on selected datasets. CorrFS and CorrBE using CR with incremental sub-routine selection and elimination respectively. Other methods are based on batch mode

Dataset	CorrFS Acc	CorrBE Acc	Gini index Acc	ReliefF Acc	Gain Acc
pima	**0.768**	0.752	0.762	0.706	*0.703*
glass	0.758	0.747	0.763	**0.765**	*0.743*
iris	*0.916*	**0.973**	0.953	0.957	0.953
soybea	0.916	0.912	**0.925**	*0.897*	0.910
Segmen	**0.985**	*0.974*	0.982	0.981	0.981
letter	0.949	*0.941*	0.957	**0.959**	0.957
satima	**0.926**	0.912	0.910	0.887	*0.880*
adult	*0.827*	0.835	0.849	**0.857**	0.853
vehicle	**0.750**	**0.750**	**0.750**	0.728	*0.703*
vowel	0.975	*0.969*	**0.979**	0.973	*0.969*
sonar	0.815	*0.801*	0.817	0.885	**0.889**
vote	0.940	0.952	**0.957**	*0.938*	0.946

	Summary				
	CorrFS	CorrBE	Gini	ReliefF	Gain
Best	4	2	4	3	1
worse	2	4	0	1	5

Table 3. Accuracy of each learning algorithms on selected datasets. Excluding on-line RF (columns 2&3) all other methods (subcolumns 5,6,7) and standard RF (column 4) are batch learning based. In summary; best and worse indicated how many times each algorithm achieved best prediction performance, worse performance respectively.

Dataset	Online RF		RF	Other methods		
	CorrFS	CorrBE		AdaB	SVM	KNN
pima	**0.771**	0.749	0.762	0.725	0.747	*0.747*
glass	0.763	0.744	0.763	**0.766**	0.762	*0.723*
iris	0.923	**0.961**	0.953	0.927	0.928	*0.898*
soy	0.922	**0.928**	0.925	**0.928**	0.918	*0.911*
Segme	0.982	*0.974*	0.982	0.986	**0.998**	0.988
letter	0.949	0.952	0.957	**0.966**	0.933	*0.893*
satim	0.918	**0.919**	0.910	0.861	0.863	*0.827*
adult	0.846	0.841	0.849	0.861	**0.877**	*0.811*
vehic	0.761	0.761	0.750	**0.765**	0.761	*0.702*
vowel	0.977	0.969	**0.979**	*0.939*	0.963	*0.939*
sonar	0.783	0.809	**0.817**	0.803	0.788	*0.761*
vote	**0.957**	0.944	**0.957**	0.952	0.953	*0.932*

	Summary					
	CorrFS	CorrBE	RF	AdaB	SVM	KNN
Best	2	3	3	4	2	0
worse	0	1	0	1	0	11

to as CorrFS and CorrBS were evaluated in both a forward and backward feature selection respectively. When comparing a pair of algorithms, we present accuracy result for each algorithm on each datasets. We estimate the success of our approach of whether if the accuracy of a learning algorithm improves or remains the same or the difference is not significant.

4. Results and Discussions

We present two sets of results corresponding to our experiments. In table 2, on 6 out of the 12 datasets considered our approach outperformed the other feature selection methods (indicated by a number in boldface in column 2 and 3). The first observation we can make from these results is that our methods compared favourably to the other feature selection methods. This is simply a demonstration of the value of correlation feature ranking in high dimensional feature spaces. In table 3, on 5 out of the 12 datasets considered there was a significant accuracy improvement due to the use of our on-line RF approach with our incremental feature selection variants (indicated by a number in boldface in column 2 and 3). With the remaining datasets accuracy achieved was competitive, or the difference was not significant. Our method performs worse only once (indicated by a number in italics in column 3). Our method yields in outperform KNN and SVM in most of dataset; this can be referred to their sensitivity of learning many irrelevance or correlated features. On the other hand, we fall short in a comparison with RF batch learning counterpart in certain cases. One of the main reasons is the ability of random forest to utilize redundant features and the diversity of growing decision trees. We are trying to justify the combination of incremental variable selection and ranking framework into on-line learning aspect. Since RF is relatively fast and stable, the focus should be in building useful selection and ranking tools toward an online ensemble classifier, yet stable and with less parameters tuning (only tree depth). Furthermore, by integrating on-line RF in our variable selection methods, search efficiency can significantly improve. Feature space induced by bootstrap replication rather than the whole training instance. Without loss of generality, the method we propose is able to learn completely in the on-line mode, and since we do not freeze the learning, it can adapt to a changing situation. Note that this kind of adaptively is not possible in the standard random forests and the ensemble classifiers.

5. Conclusions and Future Perspectives

In this paper we describe a preliminary investigation into online classification in ensemble learning domain. As we have demonstrated on-line learning and on-line feature selection is essential for a wide variety of machine learning problems. The merit of a sequential feature subset selection for the random forests is evaluated on the basis of inside into interplay in combining variable selection and ranking. In particular, the use of variable selection based on correlation ranking can be justified for on-line learning aspect to achieve relatively favourable results with respect to classification accuracy. This ability is crucial in domains with a large number of available attributes that are not all necessarily relevant to a particular classification task. Testing and evaluation show that our approach compares favorably for some datasets to the well known learning algorithms in machine learning field and to the-state-of-art feature selection methods, but mostly the difference is not significant. While the paper presents only a subset of possible applications where the proposed on-line approach can be used, further improvements are needed to explore more complex data, to collect more data, to experiment with alternative distance metrics and evaluate alternative greedy randomised attribute selection algorithms.

References

[1] Leo Breiman, "Bagging predictors," *Machine Learning*, 24(2):123.140, 1996.

[2] Leo Breiman, "Random Forests," *Machine Learning*, 45(1):5.32, 2001.

[3] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone, "Classification and regression trees," *Wadsworth Inc.*, Belmont, California, 1984.

[4] R. Schapire, Y. Freund, P. Bartlett, and W. Lee, "Boosting the margin: a new explanation for the effectiveness of voting methods," *Ann. Statist.*, 26(5):1651-1686, 1998.

[5] Marko Robnik-Sikonja, "Improving Random Forests," In J.F. Boulicaut et al.(eds): Machine Learning, ECML 2004 Proceedings, Springer, Berlin, 2004.

[6] Patrick M. Murphy and David W. Aha, UCI repository of machine learning databases, 1995. http://www.ics.uci.edu/mlearn/MLRepository.html.

[7] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artifcial Intelligence*, 97(1-2):273-324, 1997.

[8] Igor Kononenko, "Estimating attributes: analysis and extensions of Relief," Proceedings of the European conference on machine learning on Machine Learning, p.171-182. Catania, Italy, 1994.

[9] Marko Robnik-Sikonja , Koen Vanhoof, "Evaluation of ordinal attributes at value level," Data Mining and Knowledge Discovery, v.14 n.2, p.225-243, 2007

[10] M.A. Hall, L.A. Smith, "Practical feature subset selection for machine learning," *In Proceedings of the 21st Australian Compute Science Conference*, pp. 181.191, 1998

[11] Yijun Sun , Jian Li, "Iterative RELIEF for feature weighting," *Proceedings of the 23rd international conference on Machine learning*, p.913-920, Pittsburgh, Pennsylvania, 2006

[12] Marko Robnik-Sikonja , Igor Kononenko, "Theoretical and Empirical Analysis of ReliefF and RReliefF," *Machine Learning*, v.53 n.1-2, p.23-69, 2003

[13] Hiroshi Motoda , Huan Liu, "Data reduction: feature selection, Handbook of data mining and knowledge discovery," *Oxford University Press, Inc.*, New York, NY, 2002

[14] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, 3:1157-1182, 2003.

[15] Huan Liu , Hiroshi Motoda , Lei Yu, "A selective sampling approach to active feature selection," *Artificial Intelligence*, v.159 n.1-2, p.49-74, 2004

[16] Simon Perkins , Kevin Lacker , James Theiler, "Grafting: fast, incremental feature selection by gradient descent in function space," *The Journal of Machine Learning Research*, 3, 2003