Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Correlation-based Feature Ranking for Online Classification

Hassab Elgawi Osman
Imaging Science and Engineering Lab, Tokyo Institute of Technology-Japan
osman@isl.titech.ac.jp

***Abstract*—The contribution of this paper is two-fold. First, incremental feature selection based on correlation ranking (CR) is proposed for classification problems. Second, we develop online training mode using the random forests (RF) algorithm, then evaluate the performance of the combination based on the NIPS 2003 Feature Selection Challenge dataset. Results show that our approach achieves performance comparable to others batch learning algorithms, including RF.**

***Index Terms*—ensemble learning, on-line learning, feature selection, random forests, NIPS 2003.**

I. Introduction

Ensemble learning algorithms are useful in handling large feature[1] spaces even when only a small number of features is actually be useful. Randomization-based ensemble methods such as *bagging* [1], *boosting* [2], *random forests* (RF) [3], and their variants generate extra information enabling the importance of individual explanatory variables to be evaluated and substantially improving performance over single based classifiers. Most of the current ensemble learning approaches, however, work off-line or in "pseudo-on-line" batch processing, i.e., collecting a set of training examples, then running off-line ensemble algorithms. This makes ensemble learning inapplicable for many real-life machine learning problems more naturally viewed on-line than batch learning problems. In this paper, we use Breiman's random forest algorithm (RF) [3], discusses several techniques of machine learning in terms of their applicability towards on-line RF where the (random) feature subsets are generated based on two different correlation-based feature rankers (CR). We developed a new conditional permutation scheme for the computation of variable importance measure, where a considerable reduction in search is achieved, while contributing to the tradeoff between marginally lower performance and a smaller set of features necessary for learning base classifiers (here, growing decision trees to a certain depth). Our study provides similar performance to offline evaluation of other benchmark learners, such as standard RF, Adaboost, support vector machines (SVM), and K-nearest neighbors (K-NN), while expanding on our previous results [4], [5] as follows:

1) NIPS 2003 Feature Selection Challenge dataset added to an analysis. This new results should greatly interest the feature selection benchmark community.
2) Statistical analysis includes the Area Under the ROC Curve (AUC) and the balanced error rate (BER) are added. These extensions have altered some previous conclusions.
3) Statistical test results are now included in our evaluations.
4) We report the number of selected features we used compared to other methods.

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

TABLE I
Characterization of our approaches to feature selection in terms of heuristic search through the space of feature sets

Approach	Starting	Search	Halting	Induction
CorrFS	None	Greedy	Consistency	RF
CorrBE	All	Greedy	Consistency	RF

A. Our Approach

Our approach combines variable selection and variable ranking, which requires two components- a search algorithm that explores the combinatorial space of feature subsets, and one or more criterion functions that evaluate the quality of individual subsets based directly on the predictive model. Features are ranked based on correlation ranking (CR). The characterization of our approach is summarized in Table 1. The induction algorithm, RF evaluates the quality of selected feature subsets. As a search algorithm, we turn to *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 resort to an implementation that combines CR with FS and BE. We call this implementation CorrFS and CorrBE respectively. The motivation for use of this method specifically with ensembles and RF in particular stem from the inherent fast training benefit of analyzing only a few possible features. Informally, the goal of on-line RF is to minimize prediction errors and relieve the classifier of memory storage required in batch mode.

II. Background and Overview

The section that follow review review standard RF and how it estimates the importance of variables.

A. Variable Sub-selection

Variable sub-selection is guided by two general function evaluations: *filters* and *wrappers* [6]. Both distinguish between variable selection and learning, and a hybrid model has been proposed to combine their advantages [7], [8]. In another implementation, variable selection is viewed as integral process of learning [9], [10], searching through space, and generating

978-1-4244-2794-9/09/$25.00 ©2009 IEEE

potential variables based on their *relevance* to the classification problem. Many variable selection algorithms now include variable ranking for selection because of its simplicity and scalability.

B. Random Forests

Breiman's random forests (RF) is a tree-based ensemble prediction for high-dimensional classification and regression [3] that generates base decision trees using two sources of randomness bootstrap replication of instances for each tree (bagging [1]) and sampling of random subset θ 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 $\theta_1, \ldots, \theta_{k-1}$, but with the same distribution, and a tree is grown using training set T and θ_k, resulting in a classifier $h\left(x, \theta_k\right)$.

θ is used to select a subset of features. m variables are randomly sampled at each interior node using the best split with the highest Gini index. Each tree casts a unit vote for the most popular class at input random vector x with probability estimate $\hat{p} = \hat{p}(y = 1/x)$. The forest is an ensemble of tree-generated classifiers $h\left(x, \theta_k\right)$, $k = 1, \ldots, n$. For the n-th sample in 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. The larger the margin, the greater the confidence in the classification.

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

C. RF for Variable importance

The RF predictor naturally leads to numerical variable importance measures as part of its construction. Based on permutation accuracy and impurity decrease, the RF measures variable importance by randomly permuting variable m values for out-of-bag[2] (OOB) cases for tree k- if variable m is important in the classification, prediction accuracy should decrease. We can also consider reduction accumulated at nodes according to the criteria used at splits- an idea from the original CART [11] formulation. Variable importance measures are used to conduct variable selection.

D. Out-of-bag (OOB) Error Estimation

According to Breiman a forest's error rate depends on the correlation between any two trees and the strength of individual tree in the forest. One can arrive at OOB prediction as follows: for a case in original data, predict the outcome by a plurality vote using only those trees not included in their corresponding bootstrap sample. Contrasting OOB predictions with training set outcomes yield unbiased estimate of the prediction error rate for that tree- and, subsequently, for the entire forest- referred to as the OOB error rate. Despite apparent RF success [12], no potential work has been done on extending it for on-line learning, since it is difficult to design an incremental solution for importance variables. Our outcomes although simple, may provide some inspiration.

[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).

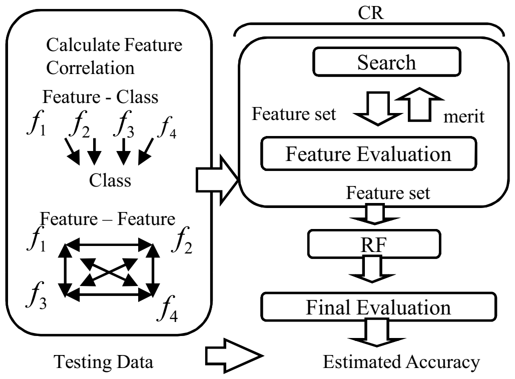

Fig. 1. CR components. Features selected by CR are passed to the RF algorithm for induction and prediction.

III. Proposed Methods

Inspired by batch learning RF success, in this section we have devised an on-line scheme for the integration of incremental selection of variable importance. Before proceeding with our presentation we need to establish some notations. Given n instances of different training set T consisting of (x_i, y_i) pairs, x_i values represent feature vectors and y_i the value to be predicted. In classification problem $y_i, i = 1, \ldots, c$ (c is the number of class values) represents one or more classes, several n bootstrap samples $(T_1, T_2, \ldots, T_n)$ are drawn from T, and an unpruned decision tree is constructed by first choosing important attribute $f(1 \leq f \leq m)$ from 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. We get correlation ranking feedback, Instead. Variables are rank based on importance scores $\omega_i x_i$ (by convention a high score is indicative for a valuable (*relevant*) feature). We write CR_j for the correlation rank of feature j and P_e for the probability of error estimate (*irrelevance*).

A. Correlation-Based Feature Ranking

Unlike filter methods, we propose ranking criteria taking into account interaction between features using *Correlation Ranking* (CR) (Fig.1) We hypothesize that relevance feature sets [6] contain features highly correlated with or predictive of class, yet mutually uncorrelated with each other.

Definition 2 (Relevant Feature) A feature F_i is said to be relevant *iff* there exists some F_i and c for which

$P(F_i = f_i) > 0$ such that $P(C = c|F_i = f_i) \neq P(C = c)$

At each step and incrementally, we estimate variable importance and, for these criteria, we have tested different search-space methods such as sequential forward feature selection (FS) and recursive backward feature elimination (BE), boiling down variable ranking problem to find a suitable measure of correlation between variables:

$$CR_j = \frac{|\,(x_j - \mu_j)^T\,(y - \mu_y)\,|}{|x_j||y|}, j = 1, 2, \ldots, D_{Feat} \tag{2}$$

where CR_j is a rank of feature j, x_j is feature vector j, y is the class label vector, μ_j and μ_y are the expectation values of feature j and class vector y respectively, and D_{Feat} is feature space dimensionality.

We first define importance score $\omega_i x_i$ from bootstrap samples by permutation of T : $\grave{T} = \{x_{i1}, \ldots, x_{ij}, \ldots, x_{in}\}$ with $\omega x_{ij} \geq \omega x_{ij+1}$; $J = 1, \ldots, n-1$. Using an efficient algorithm, we then maximize total importance scores and minimize total similarity scores of a set of features.

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

t denotes the number of selected features, $x_i = 1$ (or 0) indicates that feature f_i is selected (or not), ω_i denotes the importance score of feature f_i, and $e_{i,j}$ denotes the error estimate (*irrelevance*) between feature f_i and feature f_j. We let $e_{i,j}$ = $e_{j,i}$. Perfectly correlated variables are truly irrelevant because adding them provides no additional information. The CR algorithm is shown in Algorithm 1. The first variable selected is the variable with the smallest probability of error (P_e). The next selected produces the minimum weighted sum of P_e, and average correlation rank (ACR), etc. ACR is the mean of correlation ranking scores of the candidate variable with variables selected previously at that point. We would ordinarily continue the process until the evaluation function no longer improves, but at certain points, the change in function score is very small, because of that, in our implementation a required number of features (M) is used as the stopping criterion. In experiments, we consider ω_1 to be 0.1 and ω_2 to be 0.9.

Estimate for n samples is:

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

B. Incremental Variable Selection

Our incremental feature selection implementation performs the same sort of *incremental hill-climbing* search for generating a concept hierarchy known as *sequential selection*, which may

Algorithm 1 Correlation Ranking (CR)

```
1: Given $P_e$ + ACR $(N, M, \omega_1, \omega_2)$
2: $T = \phi$
3: Find the feature with minimum $P_e$ and append it to $T$
4: for $i = 1, 2, \cdots, M - 1$ do
5:     Find the next feature with minimum $\omega_1(p_e) + \omega_2(ACR)$
6:     Append it to $T$
7: end for
8: return $T$
```

be either *forward (FS)* or *backward (BE)*. Note that features arrive in stages after being correlation-ranked from the training data. The CorrFS algorithm (Algorithm 2) starts with no variables $F_0 = \phi$, then during each run m adds a new set $f\grave{}$ of features. The set of all features at stage m is denoted by F_m, where F_m is the union of features having just arrived with the set of features selected at stage $m-1$ at each step and greedily adding one that enhances evaluation and decreases error most, until no further addition significantly decreases error.

$$F_m = F_{m-1} \cup \{f\grave{}\}\,, \tag{5}$$

where

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

Algorithm 2 Correlation Forward Selection (CorrFS)

```
1: Calculate variable ranking $F(0)$
2: LET feature $f(0) = \phi$; error $(0) = +\infty$
3: LET feature subset $f(0) =$ all
4: for $m = 1, 2, \cdots,$ the number of all features in random
   subset do
5:     LET $f(m) = f(m-1) \cup$ the m-th best feature in $F(0)$
6:     Perform a training with $f(m)$ to obtain error rate
       $error(m)$
7: end for
8: IF $error(m) > error(m-1)$ THEN
9: terminate feature select and
10: RETURN $f(m-1)$
11: NEXT $m$
```

On the other hand CorrBE algorithm (Algorithm 3) starts with all variables $F_0 = F$ and removes $f\grave{}$ from F_0 the next worse ranked feature and trains an induction algorithm with feature set $F_0 - \{f\grave{}\}$, and $F_1 = F_0 - \{f_{max}\}$. In the next round, the algorithm tests the feature subset $F_1 - \{f\grave{}\}$ for each of the remaining features $f\grave{} \in F_1$ and removes feature f_{max} that minimizes the error of the resulting classifier to produce $F_2 = F_1 - \{f_{max}\}$. This is repeated until a local minimum of classifier error is reached- any further removal increases the error significantly- or some other stop condition is met. In practice, removing a feature may only negligibly and negatively impact on performance, so backward elimination become a tradeoff between marginally lower performance and a smaller set of features, indicating the need for some sort of regularization. Following [6], 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.

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

where

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

Algorithm 3 Backward Elimination (CorrBE)

```
1:  Calculate variable ranking $F(0)$
2:  LET $error(0)$ = average error rate;
3:  LET feature subset $f(0)$ = all
4:  for $m = 1, 2, \cdots$, the number of all features do
5:      REPEAT
6:      Let $f(m) = f(m-1)$ - the next worst feature in $F(m-1)$,
7:      Perform a training cycle with $f(m)$, calculate ranking $F$
8:      UNTIL error$(m) \leq$ error$(m-1)$ OR no more next worst feature
9:      IF no more next worst feature THEN stop feature selection
10: end for
11: RETURN $f(m-1)$
12: LET $F(m) = F$
13: NEXT $f$
```

C. On-line RF with Incremental Feature Selection

To obtain an on-line algorithm, each of the steps described in the standard RF must be on-line, where the current classifier is updated whenever a new sample arrives (Figure 2). In particular on-line RF works as follows: Based on feature ranking, we develop a new conditional permutation scheme for the computation of variable importance measure. Resulting incremental variable importance reflects the true impact of each predictor variable more reliably than the original marginal approach. According to feature ranking results, different yet random feature subsets are used as new feature spaces for learning diverse base classifiers (decision trees). Fixed set tree K is initialized, then individual trees in RF are incrementally generated by specifically selected subsets from new feature spaces. Unlike off-line RFs 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 leaf node response characterized by vector (w_i, θ_i) with $\|w_i\| = 1$. With the root node numbered 1, child nodes $2i$ and $2i + 1$ of node i are activated as follows:

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

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

where x is input data vectors, u_i represents of node i activation, and $f(.)$ is chosen as a sigmoidal function. Considering a sigmoidal activation function $f(.)$, the sum of all leaf node activation is always unity, provided that the root node has unit activation.

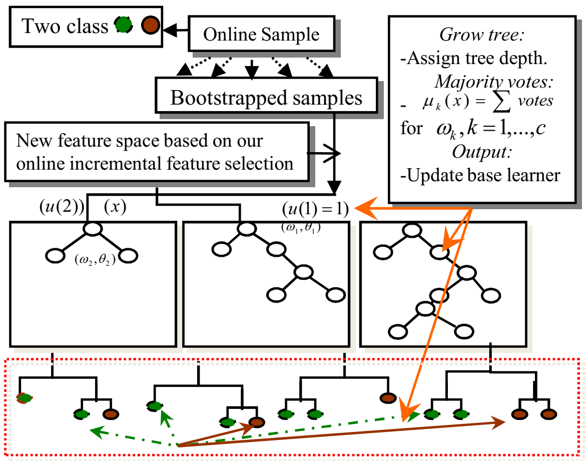

Fig. 2. On-line forest. Individual intermediate nodes accept input pattern vector x as input, and activates its two child nodes differentially based on the embedded sigmoidal function defined by its parameters (w, θ). Root node activation $u(1)$ is always unity.

D. Performance of On-line RF

There are two parameters controlling the learning classification process: The depth of the tree, and the least node. 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, creating our on-line forest by iteratively increasing its depth. We have this algorithm stop when the relative difference between the likelihood computed in two consecutive iterations does not change by 0.0001. This constrain is justified by the fact that this algorithm tends to converge asymptotically. We found that the number of trees needed for good performance eventually tails off as new data vectors are considered. Since, after a certain depth, the performance of on-line forest does not vary to a great extent, the user may choose K (the number of trees in the forest) to be a fixed value or let it to grow to the maximum possible which is at most $|T| / N_k$, where N_k is the user-chosen number of the size of each decision tree.

E. Detection Instances

To classify a new instance, we estimate the average margin of trees in instances most similar to the new instance and, after discarding trees with a negative margin, weight the tree's votes with the margin, then update the set of classifiers. Any on-line learning algorithm can update tree-base classifiers but we update for the importance of the current sample and return a new hypothesis updated with a new sample. Only one sample is used to update all base classifiers and the corresponding voting weight.

IV. Empirical Evaluation

A. Datasets

We evaluated using 5 publicly available datasets, all from NIPS 2003 challenge datasets [13], and their characteristics are listed in Table II which shows the domain each dataset it was taken from, its type (dense, sparse, or sparse binary), the number of features, the percentage of probes[3], and the number of examples in the training, validation, and test sets. All problems are two-class classification. Each missing value of feature f is treated as equal to median $\hat{f}$ of f of the training set.

B. Protocol

Classifier performance in these feature selection experiments is measured by a statistic known as the Area Under the ROC Curve (AUC) and by the balanced error rate(BER). AUC measures of classifier performance that is independent of the threshold, meaning it summarizes how true positive and false positive rates change as the threshold gradually increases from 0.0 to 1.0, i.e., it does not summarize accuracy. An ideal perfect classifier has an AUC of 1.0 and a random classifier has an AUC of 0.5. Because decision trees of a certain depth for some datasets, e.g., Dorothea may be unbalanced due to data distribution, another metric evaluation -BER- is added, i.e., the average of the error rate of the positive class and the error rate of the negative class. For classifiers with binary outputs, $BER = 1 - AUC$.

C. Results and Discussion

Results on the datasets are reported by different well known machine learning algorithms; standard (RF), AdaBoost, SVM, and KNN. To implement of AdaBoost and KNN, we use Weka library of classifier [14]. The two variants of our on-line feature selection algorithms, referred to as CorrFS and CorrBS were evaluated both in forward and backward feature selection and algorithm performance was measured after standard stratified 10-fold cross-validation. The dataset is partitioned 10-fold with the same class distribution and all folds but the i-th are used to iteratively train a classifier and the error rate of this classifier is recorded on the remaining i-th fold. This validation is repeated 10 times to avoid overfitting. Table II I presents the BER on the datasets for all learning algorithms. It shows that no method is the best on all datasets. Comparing results for CorrFS and CorrBE with the standard RF in Table III, CorrFS mostly outperforms standard RF and falls short only once for the DOROTHEA dataset. In 3 of 5 datasets considered, CorrBE outperforms the standard RF, showing that our algorithms compared favorably to other machine learning algorithms. Compared to the best reported results in the NIPS 2003 feature selection challenge. Table III shows the number of selected features (in brackets). Both variants of our incremental approach select a minimal yet good sets of features for the problem in the presence of many irrelevant or redundant features. If the number of features is too large, an exhaustive search of all feature subspace is prohibitive, as there are 2^m possible combinations for m features. Our results (Table II I) confirm that a heuristic search is more realistic than an exhaustive search but may not find optimal solutions. Note that CorrFS and CorrBE select entirely different features because feature important values of candidate features are modified, sometimes drastically, according to already selected features. In AUC results, our two variants showed slight improvement over RF, SVM, and AdaBoost on the average and very significant for KNN (Table IV); this can be referred to its sensitivity in learning many irrelevances or correlated features. Another disadvantage of KNN is its lack of functionality for variable selection. We fell short in a comparison with RF in certain cases, mainly because RF can use redundant features and due to the diversity of growing decision trees.

[3]The probe is a number of features drawn at random from a distribution resembling that of the real features, but carrying no information about the class labels. Such probes have a function in performance assessment: a good feature selection algorithm should eliminate most of the probes [13].

D. Statistical Tests

We used t-test on the results for 10-fold cross-validation to testing the statistical differences of the observed difference in AUC and BER. This is the most widely used approach for this type of experiment. Results reported in Table IV (between brackets), where '+', '-', and '=' designates a statistically significant win, loss, and the difference is not significant respectively. We separate for AUC and BER results with a slash (/). For both KNN and SVM there were a great number of statistical losses to our approaches than statistical wins. We executed Wilcoxon signed-rank test at a significance of 0.05 to determine the significance of the difference from our approaches. AUC results are significant even at 0.001. Using AUC also produced a statistically significant difference between ours and other approaches. For BER results, there was a statistically significant difference between our approach and all other approaches except for RF and Adaboost. When we compared CorrFS and CorrBE, the overall best method was CorrFS, and CorrBE appeared to be advantageous only for problems with strong conditional dependencies.

E. Computational Complexity

Our approach is easy to implement, runs very fast, and learns completely on-line mode, and because we do not freeze learning feature selection and voting-weight can change over time. Note that such adaptivity is not possible in standard RFs. The main computational effort is spent on updating base classifiers, which depends on time for calculating the feature correlation. To decrease calculation time, we assume that all feature pools are the same $F_1 = F_2 = \cdots = F_M$, so we update all corresponding base classifiers only once, speeding up the process considerably while decreasing performance only slightly. CPU time required to build a tree is two minutes. Parallel time could then be on the order of 2 minutes plus communication time.

TABLE II

NIPS 2003 CHALLENGE DATASETS. FOR EACH DATASET WE SHOW THE DOMAIN IT WAS TAKEN FROM, ITS TYPE, THE NUMBER OF FEATURES, THE PERCENTAGE OF PROBES, AND THE NUMBER OF EXAMPLES IN THE TRAINING, VALIDATION, AND TEST SETS. ALL PROBLEMS ARE TWO-CLASS CLASSIFICATION PROBLEMS.

Dataset	Domain	Type	# FE	% Pr	# Tr	# Val	# Te
ARCENE (AR)	Mass Spectrometry	Dense	10000	30	100	100	700
DEXTER (DE)	Text classification	Sparse	20000	50	300	300	2000
DOROTHEA (DO)	Drug discovery	Sparse binary	100000	50	800	350	800
GISETTE (GI)	Digit recognition	Dense	5000	30	6000	1000	6500
MADELON (MA)	Artificial	Dense	500	96	2000	600	1800

TABLE III

10-FOLD CROSS VALIDATION BERS (%) OF EACH LEARNING ALGORITHMS ON THE DATASETS. THE NUMBER OF FEATURE (%) SELECTED ARE IN BRACKETS.

		Our methods			Other methods			Challenge best
Data	Baseline	CorrFS	CorrBE	RF	AdaB	SVM	KNN	$BER \pm \delta_{BER}$
AR	0.1470	11.04(15.1)	13.31(14.0)	11.25	21.43	13.31	21.43	$0.1073 \pm 0.0117(10.7)$
DE	0.0500	8.00(34.1)	6.50(22.6)	8.00	8.33	11.67	22.00	$0.0330 \pm 0.0040(18.57)$
DO	0.1237	21.38(35)	16.82(4.2)	12.51	39.38	33.98	14.21	$0.0854 \pm 0.0099(100)$
GI	0.0180	1.37(2.3)	1.37(2.8)	1.80	1.80	2.10	2.08	$0.0126 \pm 0.0014(18.32)$
MA	0.0733	13.00(14.4)	7.11(11.0)	13.00	39.85	40.17	11.60	$0.0622 \pm 0.0057(1.6)$

TABLE IV

10-FOLD CROSS VALIDATION AUC OF EACH LEARNING ALGORITHMS ON THE DATASETS. STATISTICAL RESULTS FOR EACH DATASET AT CONFIDENCE LEVEL OF 0.95 ARE IN BRACKETS.

	Our methods			Other methods		
Data	CorrFS	CorrBE	RF	AdaB	SVM	KNN
AR	84.73(+/+)	90.63(+/+)	80.76(+/=)	69.00(+/+)	72.60(+/+)	68.60(-/-)
DE	93.50(+/+)	96.86(+/+)	95.05(+/+)	81.00(+/+)	82.47(+/+)	77.40(-/-)
DO	83.50(+/+)	77.56(+/=)	86.07(=/+)	92.18(+/+)	92.22(+/=)	91.70(+/+)
GI	92.63(+/+)	86.71(+/=)	97.42(+/+)	82.25(=/+)	90.78(+/+)	84.16(-/+)
MA	93.39(+/+)	92.89(+/+)	90.56(+/+)	74.92(+/+)	85.76(+/+)	73.94(+/+)

V. CONCLUSIONS AND PROJECTED WORK

We have described on-line classification in an ensemble learning domain. As we have shown on-line learning and incremental feature selection are essential for a wide variety of machine learning problems. We have evaluated the advantage of sequential feature subset selection for random forests is evaluated on the basis of combining variable selection and ranking. The use of variable selection based on correlation ranking is justifiable for on-line learning to achieve relatively favorable results for classification accuracy. This is crucial in domains with a large number of available features not all necessarily relevant to a particular classification task. Testing and evaluation showed that our approach compares favorably for some datasets to well-known learning algorithms in machine learning, but the difference is rarely significant. While extending on some of our previous results, we also present new results greatly interest the feature selection benchmark community. We now plan experiments with possible alternative applications using our on-line approach and evaluate other greedy randomized attribute selection algorithms.

REFERENCES

[1] L. Breiman, "Bagging predictors," *Machine Learning*, 24(2):123-140, 1996.

[2] 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.

[3] L. Breiman, "Random Forests," *Machine Learning*, 45(1):5-32, 2001.

[4] H. Elgawi Osman, "Online Random Forests based on CorrFS and CorrBE," in *Proc.IEEE workshop on online classification*, (CVPR'08), pp.1-7, 2008

[5] H. Elgawi Osman, "Variable Ranking for Online Ensemble Learning,", in *Proc. The 24th Annual ACM Symposium on Applied Computing (ACM SAC)*, 2009

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

[7] H. Liu , H. Motoda , L. Yu. "A selective sampling approach to active feature selection," *Artificial Intelligence*, volume 159(1-2):49-74, 2004.

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

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

[10] S. Perkins , K. Lacker , and J. Theiler. "Grafting: fast, incremental feature selection by gradient descent in function space," *J. Machine Learning Research*, 3:1333-1356, 2003.

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

[12] R. E. Banfield, L. O. Hall, K. W. Bowyer, D. Bhadoria, W. P. Kegelmeyer and S. Eschrich, "A comparison of ensemble creation techniques," in *Proc. International Conference on Multiple Classifier Systems*, 2004.

[13] I. Guyon, "Design of experiments of the NIPS 2003 variable selection benchmark," http://www.nipsfsc.ecs.soton.ac.uk/papers/Datasets.pdf, 2003.

[14] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools with Java Implementations*. Morgan Kaufmann, San Francisco, 1999.