Calculating Feature Importance in Data Streams with Concept Drift using Online Random Forest

Andrew Phelps Cassidy
Commonwealth Computer Research Inc. (CCRi)
Charlottesville, USA
andrew.cassidy@ccri.com

Frank A. Deviney Jr., Phd
Commonwealth Computer Research Inc. (CCRi)
Charlottesville, USA
frank.deviney@ccri.com

***Abstract*—Large volume data streams with concept drift have garnered a great deal of attention in the machine learning community. Numerous researchers have proposed online learning algorithms that train iteratively from new observations, and provide continuously relevant predictions. Compared to previous offline, or sliding window approaches, these algorithms have shown better predictive performance, rapid detection of, and adaptation to, concept drift, and increased scalability to high volume or high velocity data. Online Random Forest (ORF) is one such approach to streaming classification problems. We adapted the feature importance metrics of Mean Decrease in Accuracy (MDA) and Mean Decrease in Gini Impurity (MDG), both originally designed for offline Random Forest, to Online Random Forest so that they evolve with time and concept drift. Our work is novel in that previous streaming models have not provided any measures of feature importance. We experimentally tested our Online Random Forest versions of feature importance against their offline counterparts, and concluded that our approach to tracking the underlying drifting concepts in a simulated data stream is valid.**

Keywords—Feature Importance; Online Random Forest; Concept Drift; Data Streams

I. INTRODUCTION

A. Problem Description

A data stream is a dataset where observations have an explicit arrival order. The canonical example is a dataset ordered by the arrival time of the observations. In addition, new observations can arrive continuously in high volume and velocity [1]. Data streams pose a unique set of problems to machine learning researchers and practitioners. One of the most difficult problems is concept drift. To be useful, models must adapt to concept drift, yet not overreact to process noise [1].

We define concept drift to mean a temporally indexed change in the process of interest. For example, a model that classifies emails as Spam or Ham (a legitimate email) based on the presence of keywords, characters, or topics can become invalid due to a shift in how spammers construct messages [2].

Initial approaches to learning from data streams with concept drift were based on explicitly detecting, quantifying, and characterizing concept drift and then either re-retraining or re-weighting classifiers [3]. Other approaches relied on periodically creating new models, each trained on a different sliding time window of data [4]. Recently proposed alternatives instead learn incrementally [5][6]. We focus on an incremental learning algorithm for classification problems: Online Random Forest (ORF) [7].

ORF, and many other streaming algorithms, learn incrementally on each new observation and restructure based on temporally indexed predictive performance. The objective is better online predictive performance, rapid detection of, and adaptation to, concept drift, and increased scalability to high volume or high velocity data. We begin with a brief discussion of the original Random Forest algorithm, and then discuss adapting it to the streaming realm. Finally, we discuss our contribution of feature importance measures to ORF, and demonstrate our ability to track and visualize concept drift through experimental results.

B. Online Random Forest

The original Random Forest (RF), first proposed by Leo Breiman in 2001, has been used extensively for classification and regression problems, ranging from fields as diverse as computer vision, to genomics [8] [9]. Random Forest is an offline, or batch, ensemble method that creates a number of decision trees, each trained on a different bootstrapped aggregation of the training set. In addition, for each decision split created in a tree, only a subset of the total number of predictors is considered for splitting the data. This helps to prevent overfitting and encourages diversity in what each tree learns. Individual classifications by each tree are aggregated to yield one classification.

Saffari, Leistner, Santner, Godec, and Horst [7] proposed ORF in 2009, and expanded on offline Random Forest by adding:

- Incremental Tree Building
- Online Bootstrap Aggregation
- Tree Discarding

Each of these properties makes ORF robust to concept drift without explicit signaling from an outside monitoring system, and without having to window data and iteratively re-train new models.

1) Incremental Tree Building

As data streams in, ORF periodically generates new split decisions in tree nodes. Statistics about which feature best splits the data, and when to split the data, are calculated over time. The split decision is made when 1) the node has seen a minimum number of observations, and 2) the intended split satisfies a model improvement adequacy rule [7]. Once the split is made, additional observations presented to the tree cannot change the rule governing the split; instead each new observation falls through to a child node.

Incremental tree building enables adaptive learning. When making an incremental decision on how to split streaming data, ORF considers a subset of the entire population of predictors as candidates for splitting. This subset of predictors could include variables heretofore deemed unimportant. A tree could, therefore, split on one concept at the top of the tree and a different concept at the bottom of the tree.

2) Online Bootstrap Aggregation (Bagging)

Bootstrap Aggregation (also called bagging) of data in an offline RF means sampling with replacement from data, to create a unique dataset for each tree in the ensemble [10]. Bagged data is separated into training and testing (also called Out of Bag (OOB)) datasets. ORF bags and separates stream observations for training and testing online using random samples from a Poisson distribution [11]. A tree trains on the training stream and uses the testing stream to estimate its accuracy and error rates.

3) Tree Discarding

As concepts change, trees trained on old concepts will incorrectly classify testing stream observations and incur degraded accuracy. To adapt, ORF discards underperforming trees using a random probability function based on a tree's test stream error rate [7]. Underperforming trees are replaced with untrained trees ("stumps"), better able to learn new concepts. Tree discarding is a form of online adaptation and ensemble restructuring.

In the next section we present our main contribution: the adaptation of two offline feature importance measures to the online setting for ORF classification models. In section III, we test our algorithms' success on the benchmark hyperplane dataset and compare its success to that of a sliding window approach. In section IV we conclude the paper.

II. Methods

Our contribution to ORF is the adaptation of offline feature importance measures to the online setting for classification models. We have implemented two methods of feature importance for Online Random Forest classifiers: Mean Decrease in Gini Impurity (MDG) and Mean Decrease in Accuracy (MDA).

A. Mean Decrease in Gini Impurity (MDG)

A tree is made up of a hierarchy of nodes. A node takes incoming observations and sends them to its left or right child node according to a split rule. In doing so, a node attempts to purify the incoming dataset. A perfectly pure dataset is one that only contains one class of observations. Gini Impurity (1) is a measure of a dataset's deviation from being perfectly pure.

$$I(dataset) = \sum_{i=0}^{numClasses} (p_i)(1-p_i) \qquad (1)$$

where p_i = proportion of class i in dataset

Decrease in Gini Impurity (2) measures the change in impurity at a node caused by sending incoming observations left or right. A high Decrease in Gini Impurity implies that the node and its corresponding feature were useful at separating the classes of interest.

$$\Delta I(Node) = I(d_{incoming}) - P_L I(d_L) - P_R I(d_R) \qquad (2)$$

where $d_{incoming}$ = incoming dataset, $d_{L,R}$ = dataset sent left or right, n = number of observations in incoming dataset, $n_{L,R}$ = number of observations in left or right dataset, $P_L = n_L/n$, and $P_R = n_R/n$

Mean Decrease in Gini Impurity (MDG) averages the change in impurity seen across all nodes that used the same feature. This metric measures the forest-wide contribution of the feature at separating the different classes and constitutes one measure of feature importance.

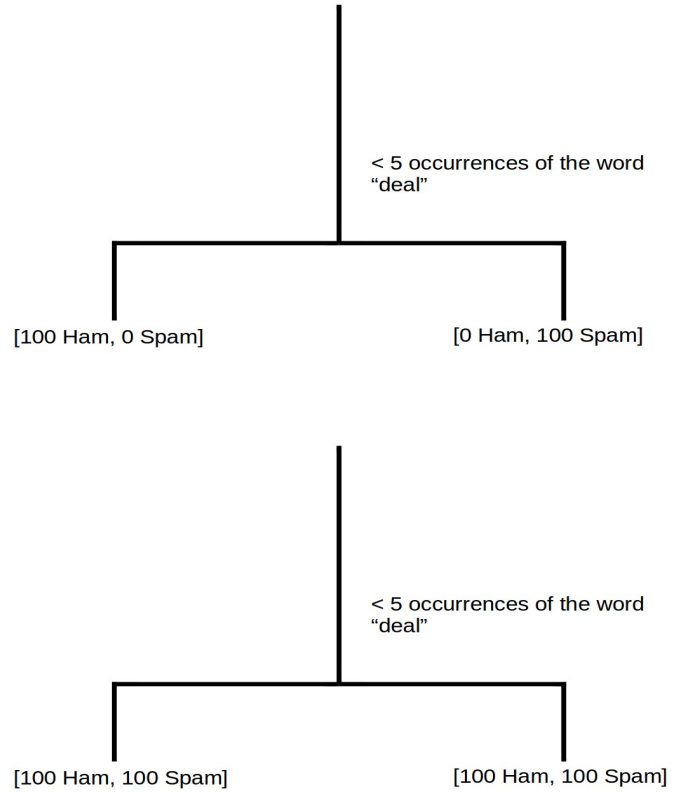

Fig. 1. The top node splits observations using the feature "occurrence of the word deal" with a feature value of "5 or more occurrences" and has the max Decrease in Gini Impurity of 0.5. The bottom node is the same node after training on 200 more observations. The node now does nothing to purify the incoming data and has a Decrease in Gini Impurity of 0.

In ORF, nodes' Decrease in Gini Impurities evolve with concept drift as new observations are streamed (Fig. 1). Furthermore, if a node is no longer useful at purifying data, the tree associated with the node will incur degraded performance on test observations. In that case, the forest is more likely to discard the tree, thus eliminating the tree's contribution to forest-wide MDG. These two traits, in combination, make MDG robust to concept drift.

B. Mean Decrease in Accuracy (MDA)

The classification results (predicted class vs. actual class) of test observations form a tree's confusion matrix, from which accuracy is calculated. Accuracy, which serves as an unbiased estimate of a tree's performance, is used as a measure for both determining whether or not to discard a tree, and to calculate another measure of feature importance: Mean Decrease in Accuracy.

A feature's Mean Decrease in Accuracy (MDA) is the average change in accuracy across the forest, resulting from permuting the feature's values in test observations. Permuting a feature's value entails substituting a different observation's value for a test observation's true value (e.g., the true value of "four" occurrences of the word "deal" is substituted with the random value "eight" occurrences of the word deal; this value "eight" is sampled from a random email). The classification results from each feature's permuted test observations form a new confusion matrix for each tree. Permuted accuracy for each feature is calculated from its corresponding confusion matrix.

The change in accuracy (without permutation minus with permutation) is then calculated for each feature, for each tree, and averaged across the forest. This average (MDA) measures the forest wide contribution of the feature at predicting the different classes and constitutes another measure of feature importance. If permutation has no effect on accuracy (i.e. a MDA close to 0), one can conclude that the feature was not important. If that permutation does have an effect, the change in accuracy is interpreted as the importance of that variable.

Adapting MDA to ORF required a complete retooling of the underlying MDA calculations for two reasons:

- In a stream, there is no stored buffer of observations to sample alternative features values during permutation
- Test observations are streamed and accuracy is constantly updated

1) Alternative to a buffer of observations

To overcome not having a buffer of observations, we used a technique developed by offline Random Forest researchers dealing with missing data in test observations. In Hapfelmeir, Hothorn, Ulm, and Strobl's method [12], instead of permuting a feature's test observations' values, the observations are randomly allocated to the left or right nodes, if the parent node's feature is the feature of interest. The allocation to left or right node is determined probabilistically by sampling left or right randomly from the relative frequency (3) of left and right training observations.

$$P_L = n_L / n \qquad P_R = n_R / n \tag{3}$$

where $n_{L.R}$ = number of observations in left or right node and n = number of observations in parent node

Across 1000 trials on a simulated dataset, Hapfelmeir, Hothorn, Ulm, and Strobl [12] concluded that their new probabilistic approach and the original permutation approach were approximately equal to one another when no data was missing, and they put forth that the slight deviations were due to the inherent variability of the importance measures. Because their approach was experimentally robust and could be performed online without a stored buffer of observations, we use it as a technique for MDA in ORF.

2) Online Confusion Matrices

ORF designates each observation of its stream as either a training observation or a test observation for each tree in the ensemble. Each tree has it's own online confusion matrix (used for tree accuracy) along with an online confusion matrix for each feature (used for permuted accuracy). These online confusion matrices each update on the classification result of the un-permuted test observation or the probabilistically permuted test observation (Fig. 2).

The continuous testing process of ORF allows a feature's Mean Decrease in Accuracy to evolve with time. For example, if concept drift occurs such that a feature is no longer predictive of the classes of interest, the feature's MDA will start to decrease as the trees that use that feature in splits are tested on the new concepts. This is particularly true for older, larger trees that were built when an earlier concept was in effect. Subsequently, the forest will remove trees that exhibit degraded accuracy due to splitting on "no longer" important features. These two traits, in combination, make MDA robust to concept drift.

I. Update the confusion matrix of the tree with the classification result of the test observation

II. For each feature:

a. Classify the test observation, but randomly assign the observation a left or right child node if the observation arrives at a node with the feature of interest

b. Update the feature's confusion matrix with the classification result

III. Compute accuracy using the tree's confusion matrix

IV. For each feature:

a. Compute accuracy using the feature's permuted confusion matrix

b. Compute the difference between the tree's accuracy and the permuted accuracy

V. Repeat steps I through IV for each tree and use the mean difference over all trees as the overall importance score [12]

Fig. 2. Pseudocode for a new test observation

III. Experiment

A. Hyperplane Description

One of the most popular simulated datasets used to test performance of streaming algorithms [13] [14] is the rotating hyperplane dataset first used by Hulten, Spencer, and Domingo [6]. To simulate gradual drift in the importance of different predictors we use a d-dimensional hyperplane (4).

$$\sum_{i=0}^{d} a_i x_i = a_o \qquad (4)$$

where each x_i and a_i is a sample from a uniform distribution with bounds [0,1]

Thus, each x vector is a point in a d-dimensional space and the corresponding a's are the weights of each dimension. Vectors that satisfy (5) are labeled Class One and vectors and satisfy (6) are labeled Class Two. a_o (7) is chosen such that the hyperplane cuts the multi-dimensional space in two parts of the same volume; therefore, half of the examples are labeled Class One and the other half are labeled Class Two. In addition to the strict labeling of Class One and Class Two there is a 5% chance that a class will be labeled incorrectly.

$$\sum_{i=1}^{d} a_i x_i \leq a_o \qquad (5)$$

$$\sum_{i=1}^{d} a_i x_i > a_o \qquad (6)$$

$$a_o = \frac{1}{2}\sum_{i=1}^{d} a_i x_i \qquad (7)$$

Three parameters control the gradual drift of the weights on each predictor:

- K specifies the number of dimensions whose corresponding weight will drift
- T specifies the magnitude of the change in weight for each new observation
- S specifies the probability that the direction of change is reversed.

Also, for each new observation generated a_o is recomputed so that the class distribution remains equally likely for Class One and Class Two.

B. Hyperplane Measure of Success

The magnitude of a dimension's weight measures its uniform random sample's contribution to the ultimate label for the observation. The sign (+, -) of the weight does not matter because the sign is considered in the calculation for a_o (the threshold for flagging Class One vs. Class Two). Said another way, the absolute magnitude of a weight on a dimension can be considered the true feature importance of that dimension. In addition to a dataset, our simulation of the hyperplane outputs the time series of weights for drifting dimensions. Our measure of success entailed quantifying how well our feature importance metrics aligned with the weights of drifting dimensions.

For our metric, we averaged the correlations of drifting features' absolute value of the dimension weight time series and the associated feature importance time series. We compared the ORF's MDA and MDG correlation metrics to the MDA and MDG correlation metrics from a sliding window approach using the randomForest package version 4.6-7 in R version 3.0.2 [15] [16].

C. Hyperplane Experiment

We generated 30 ten-dimensional datasets (d=10) of 10,000 observations each. 15 of the datasets used Parameter Set 1 (K=2, T=1, S=0.10), and the other 15 used Parameter Set 2 (K=5, T=1, S=0.10).

For each dataset, we created a different ORF and trained it incrementally on each ordered set of observations. Every 10 observations after the first 100 observations, we made requests for MDA and MDG.

For each dataset, we also created 990 different offline Random Forest models trained on a window of 100 observations with a slide of 10 observations. Each individual model reported MDA and MDG. We then aggregated the results from MDA and MDG for ORF and offline RF into separate time series of length 990 (Fig. 3 & Fig. 4).

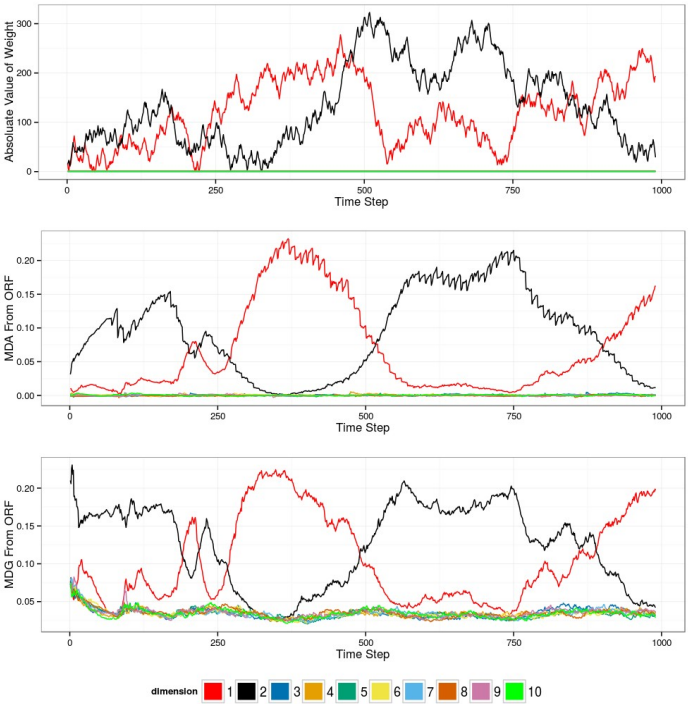

Fig. 3. Visualization of absolute value of dimension weights, MDA, and MDG time series from one trial of ORF on Parameter Set 1 (K=2, T=1, S=0.10) – Only dimensions 1 and 2, which correspond to K=2, show any drift in their weights. As well, dimensions 1 and 2 are the only dimensions that fluctuate significantly with gradual change in feature importance.

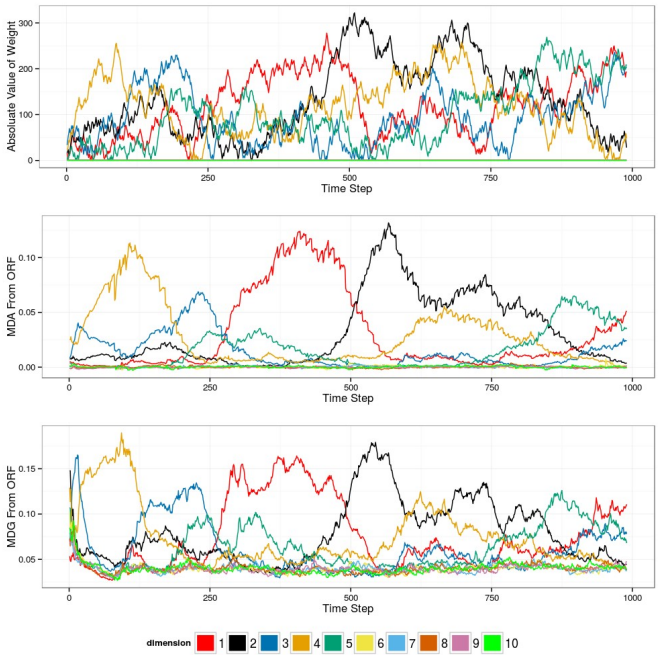

Fig. 4. Visualization of absolute value of dimension weights, MDA, and MDG time series from one trial of ORF on Parameter Set 2 (K=5, T=1, S=0.10) – Only dimensions 1 through 5, which correspond to K=5, show any drift in their weights. As well, dimensions 1 through 5 are the only dimensions that fluctuate significantly with gradual change in feature importance.

D. Hyperplane Results

1) ANOVA

To compare the 4 different feature importance measures' correlation metric scores across the 30 different trials we performed a repeated measures analysis of variance (ANOVA) test (8). The repeated measures ANOVA test controls for the variation between the treatments' (feature importance measures) performance on different datasets.

$$H_o : \mu_{OFT-MDA} = \mu_{ORF-MDG} = \mu_{MDA} = \mu_{MDG} \quad (8)$$

$$H_\alpha : \text{ at least one differs}$$

where μ = average correlation metric score across 30 trials

TABLE I. THE OUTPUT OF THE REPEATED MEASURES ANOVA TEST

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Residuals	29	3.37	0.12		
measure	3	0.03	0.01	7.69	0.0001
Residuals1	87	0.12	0.00		

The p-value associated with our ANOVA test is 0.0001. Therefore, under an α of .05, we rejected the null hypothesis that the population means for the different feature importance measures are the same. As a post-hoc test we performed pairwise t-tests (9) between the different treatments and adjusted for multiple comparisons using bonferroni corrections.

2) Pairwise T-Tests

$$H_0 : \mu_1 - \mu_2 = 0$$
$$H_\alpha : \mu_1 - \mu_2 > 0 \quad (9)$$

where μ = average correlation metric score

TABLE II. P-VALUES ASSOCIATED WITH THE PAIRWISE TESTS

	MDA	MDG	ORF MDA
MDG	1.0000		
ORF MDA	0.2434	0.0065	
ORF MDG	0.0898	0.0004	1.0000

a. Row labels correspond to μ_1 and column labels correspond to μ_2

We rejected the null hypothesis for the comparison of ORF-MDA and MDG and ORF-MDG and MDG. We failed to reject the null hypothesis for ORF-MDA and MDA and ORF-MDG and MDA under an α level of .05. We concluded that ORF's MDA and MDG methods statistically either outperformed or were on par with the offline methods used in the sliding window approach in terms of correlating with the true drift of the dimensions' weights.

IV. DISCUSSION AND CONCLUSIONS

We adapted the feature importance methods of Mean Decrease in Accuracy and Mean Decrease in Gini Impurity to Online Random Forest. The stream learning properties of ORF allow the adapted feature importance metrics of MDG and MDA to respond to concept drift in the importance of different predictors towards the classification problem. Through experimental results we demonstrated that ORF's adapted feature importance methods of MDA and MDG outperformed, or were on par with, sliding window traditional feature importance methods on tracking the true feature importance of dimensions in simulated hyperplane datasets. In future research, we hope to measure ORF's adapted methods' success on different styles of drift including: step and blip style drift [3]. We hypothesize that even greater performance gains may be found in the presence of step-style drift. In general, we hope to introduce the machine learning community to the idea of on-demand feature importance reports to accompany on-demand training and classification.

REFERENCES

[1] F. Chu, Y. Wang, and C. Zaniolo, "Mining Noisy Data Streams via a Discriminative Model," presented at the 7th International Conference. DS, Padova, Italy, 2004, vol. 3245, pp. 47–59.

[2] S. J. Delany, P. Cunningham, A. Tsymbal, and L. Coyle, "A Case-Based Technique for Tracking Concept Drift in Spam Filtering," Knowledge Based Systems, vol. 18, no. 4–5, pp. 187–195, Aug. 2005.

[3] L. I. Kuncheva, "Classifier ensembles for detecting concept change in streaming data: Overview and perspectives," in Proceedings of the 2nd Workshop SUEMA, 2008, vol. 2008, pp. 5–10.

[4] I. Zliobaite and L. I. Kuncheva, "Determining the training window for small sample size classification with concept drift," presented at the IEEE International Conference on Data Mining Workshops, 2009, pp. 447–452.

[5] R. Polikar, L. Udpa, S. Udpa, and V. Honavar, "Learn++: An Incremental Learning Algorithm for Supervised Neural Networks," IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS —PART C: APPLICATIONS AND REVIEWS, vol. 31, no. 4, pp. 497–508, Nov. 2001.

[6] G. Hulten, L. Spencer, and P. Domingo, “Mining time-changing data streams,” presented at the 7Th International Conference on Knowledge Discovery and Data Mining, 2001, pp. 97–106.
[7] A. Saffari, C. Leistner, J. Santner, M. Godec, and B. Horst, “On-line Random Forests,” presented at the 3rd IEEE ICCV Workshop on On-Line Learning for Computer Vision, 2009.
[8] L. Breiman, “Random Forest,” Machine Learning, vol. 45, no. 1, pp. 5–32, Oct. 2001.
[9] X. Chen and H. Ishwaran, “Random forests for genomic data analysis,” Genomics, vol. 99, no. 6, pp. 323–329, Jun. 2012.
[10] L. Breiman, “Bagging Predictors,” Machine Learning, vol. 24, no. 2, pp. 123–140, Aug. 1996.
[11] N. C. Oza, “Online Bagging and Boosting,” presented at the IEEE Conference on Systems, Man and Cybernetics, 2005, vol. 3, pp. 2340–2345.
[12] A. Hapfelmeier, T. Hothorn, K. Ulm, and C. Strobl, “A new variable importance measure for random forests with missing data,” Statistics and Computing, vol. 24, no. 1, pp. 21–34, Jan. 2014.
[13] J. Z. Kolter and M. A. Maloof, “Dynamic Weighted Majority: An Ensemble Method for Drifting Concepts,” presented at the Third IEEE International Conference on Data Mining, 2003, pp. 123–130.
[14] H. Wang, W. Fan, P. S. Yu, and J. Han , “Mining Concept-Drifting Data Streams using Ensemble Classifiers ,” presented at the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2003, pp. 226–235.
[15] R Core Team (2013). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.
[16] A. Liaw and M. Wiener, “Classification and Regression by randomForest,” R News, vol. 2, no. 3, pp. 18–22, 2002.