Contemporary Engineering Sciences, Vol. 9, 2016, no. 32, 1547 - 1557
HIKARI Ltd, www.m-hikari.com
https://doi.org/10.12988/ces.2016.69153

A Novel Hybrid Partial Least Squares Quadratic Discriminant Classification Algorithm

Kyaw Kyaw Htike

School of Information Technology
UCSI University
Kuala Lumpur. Malaysia

Abstract

Giving computers and machines the ability to recognize and classify patterns is an important task in engineering systems with numerous applications in many aspects of life. The field of pattern classification has become very popular in the past decade and although numerous techniques have been proposed, there is still much room for improvement. In this paper, we propose a novel pattern classification algorithm which we term "Hybrid Partial Least Squares Quadratic Discriminant Classifier" that outperforms several state-of-the-art classifiers. Evaluating on three publicly available challenging datasets demonstrates the effectiveness of our proposed approach.

Keywords: classification, pattern recognition, hybrid, machine learning

1 Introduction

Pattern classification is a crucial sub-area under Machine Learning that has far-reaching uses in fields such as Computer Vision, Natural Language Processing and Speech Processing. Given an unseen or novel pattern, the goal is to classify or categorize it to one of the pre-defined list of categories or classes.

The main challenge behind pattern classification is building up of a function that can generalize well to any unseen future input data. Normally, rather than attempting to manually handcraft rules by humans, the most successful

approach has been found to be adopting a Machine Learning approach whereby input-output pairs of data are gathered, a model with a certain set of *latent* (or hidden) parameters is constructed and then a learning algorithm is used to optimize the parameters of the model with respect to the data. Although this still requires effort from humans (in the form of gathering sufficient quantity and quality of data), this is still relatively easier than having to manually write rules.

A successful pattern classification algorithm can be used in many applications such as face detection [1, 2, 3], voice recognition [4], pedestrian detection [5, 6, 7], sound classification [8], text localization [9] and human pose estimation [10, 11].

There have been many classifiers proposed in the literature. Some of the most widely used state-of-the-art ones are Support Vector Machines (SVMs) [12, 13] with both linear kernel and non-linear kernels, Random Forests [14] and variations of AdaBoost [15] such as the Discrete AdaBoost [16] and the Real AdaBoost [17].

SVMs with linear kernels [12] have the limitation of not being able to model non-linear decision boundaries. This was overcome with the invention of non-linear kernels [13]; however, they are often slow to train and inefficient at test time due to the majority of the training data points ending up at *support vectors*, especially when the input feature vectors are high dimensional. In contrast, our proposed method is very efficient both at training and test time while at the same time having the ability to model a subset of non-linear decision boundaries, namely quadratic surfaces.

Random Forest [14], proposed by Breiman, is a widely used state-of-the-art algorithm that integrates the concepts of bagging [18] and random subspace [19] to decision trees. However, similar to other ensemble methods, it has the disadvantage of having to train multiple classifiers (*i.e.* trees in this case) which makes the overall classifier to be highly redundant, to require a large storage space and to be inefficient at training and test times. AdaBoost and most of the its variants [16, 17] share similar limitations and are also known for their overfitting on many datasets. In contrast, our proposed approach results in a very compact set of projections which minimizes the aforementioned problems.

Unlike non-linear SVMs, Random Forests and AdaBoost which are meant to model arbitrary non-linear decision boundaries, our proposed classifier restricts the set of learnable decision boundaries to quadratic surfaces which are highly useful in many datasets especially high dimensional ones. In addition, for many problems (especially in high dimensional spaces), a quadratic surface in the feature space is usually sufficiently non-linear to separate the given classes.

Our proposed approach is termed as "Hybrid Partial Least Squares Quadratic Discriminant Classifier" (PLSQuad) which integrates and extends Partial Least Squares (PLS) Regression [20] and Quadratic Discriminant Analysis [21]. Al-

though the proposed method can be used for multi-class classification, we focus on binary classification to simplify the formulation and for improved clarity. The formulation can easily be extended to multi-class problems by using the algorithm in this paper.

2 Method

Assume that the training dataset is given by:

$$\mathcal{D} = \{(\mathbf{x}^1, y^1), , \ldots, (\mathbf{x}^N, y^N)\} \tag{1}$$

where $\mathbf{x}^i \in \mathrm{R}^d$ is the d-dimensional input feature vector corresponding to the i-th training data point, and $y^i \in \{1, 0\}$ the corresponding output label which can take either the values 1 or 0 for the first class and the second class respectively (since we are only considering binary classification in the formulation).

The goal of the training process is to learn a mapping function f such that $y = f(\mathbf{x}, \Theta)$ where f is the *classifier* (*i.e.* the classification function) and Θ is the set of latent parameters that characterize f and that need to be optimized during the training process. The goal of the training process then can be thought of as finding the optimal values for Θ that makes f perform well on the training data $\mathcal{D}$, with the ultimate objective of f being able to generalize well to $\mathbf{x}$ that it has never seen before (*i.e.* not a member of $\mathcal{D}$). This never-seen-before data is often called the *test* data.

The algorithm for training the classifier f is given in Algorithm 1. The inputs to the algorithm are the training dataset $\mathcal{D}$ and the number of projections z which is a hyper-parameter of the training algorithm and its optimal value can be automatically found using cross-validation. We describe the algorithm in detail below.

Firstly, we form the matrix $\mathbf{A}$ which contains the input feature vectors $\mathbf{x}^i \in \mathcal{D}$ as rows. Furthermore, a vector $\mathbf{b}$ is constructed from $[y^1, y^2, \ldots, y^N]$. We then initialize $\mathbf{A}_0$ to $\mathbf{A}$, *i.e.* $\mathbf{A}_0 \leftarrow \mathbf{A}$.

We also initialize $\mathbf{w}_0$ and t_0 as follows:

$$\mathbf{w}_0 \leftarrow \mathbf{A}^T\mathbf{b} / \left\|\mathbf{A}^T\mathbf{b}\right\| \tag{2}$$

$$t_0 \leftarrow \mathbf{A}w_0 \tag{3}$$

We then enter a loop k which is processed for $k = 1$ to $k = z$ iterations. In each of the iterations, we update t_k, p_k and q_k as follows:

$$h \leftarrow t_k^T t_k \tag{4}$$

Algorithm 1 The proposed classifier training algorithm

Input: Training dataset $\mathcal{D}$, number of projections z
Output: Classifier f

$[\mathbf{A}, \mathbf{b}] \leftarrow \mathcal{D}$
$\mathbf{A}_0 \leftarrow \mathbf{A}$
$\mathbf{w}_0 \leftarrow \mathbf{A}^T\mathbf{b} / \left\|\mathbf{A}^T\mathbf{b}\right\|$
$t_0 \leftarrow \mathbf{A}w_0$
for $k = 1$ **to** z **do**
  $h \leftarrow t_k^T t_k$
  $t_k \leftarrow t_k / h$
  $p_k \leftarrow \mathbf{X}_k^T t_k$
  $q_k \leftarrow \mathbf{b}^T t_k$
  if $q_k = 0$ **then**
    $z \leftarrow k$
    break
  end if
  if $k < z$ **then**
    $\mathbf{A}_{k+1} \leftarrow \mathbf{A}_k - h t_k p_k^T$
    $\mathbf{w}_{k+1} \leftarrow \mathbf{A}_{k+1}^T \mathbf{b}$
    $t_k \leftarrow \mathbf{A}_{k+1}\mathbf{w}_{k+1}$
  end if
end for
$\mathbf{W} \leftarrow \left[\mathbf{w}_0, \mathbf{w}_1, \ldots, \mathbf{w}_{z-1}\right]$
$\mathbf{A} \leftarrow \mathbf{AW}$
Let $g(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^k |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})\right)$
Let $s(\mathbf{A}, n)$ be the function to generate a new matrix by selecting rows from $\mathbf{A}$ that correspond to $\mathbf{b}_i = n$
$\mu_{y=1} \leftarrow$ mean of vectors from $s(\mathbf{A}, 1)$
$\mu_{y=0} \leftarrow$ mean of vectors from $s(\mathbf{A}, 0)$
$\Sigma_{y=1} \leftarrow$ covariance of vectors from $s(\mathbf{A}, 0)$
$\Sigma_{y=0} \leftarrow$ covariance of vectors from $s(\mathbf{A}, 0)$
Let $\Theta = \{\mu_{y=1}, \Sigma_{y=1}, \mu_{y=0}, \Sigma_{y=0}\}$

$$f(\mathbf{x}, \Theta) = \frac{g(\mathbf{x}, \mu_{y=1}, \Sigma_{y=1})}{g(\mathbf{x}, \mu_{y=0}, \Sigma_{y=0})} < \text{thresh}$$

return f

$$t_k \leftarrow t_k/h \tag{5}$$

$$p_k \leftarrow \mathbf{X}_k^T t_k \tag{6}$$

$$q_k \leftarrow \mathbf{b}^T t_k \tag{7}$$

If $q_k = 0$, we set z to k and break out of the loop prematurely. Otherwise, we update the $\mathbf{A}$, $\mathbf{w}$ and t_k as:

$$\mathbf{A}_{k+1} \leftarrow \mathbf{A}_k - h t_k p_k^T \tag{8}$$

$$\mathbf{w}_{k+1} \leftarrow \mathbf{A}_{k+1}^T \mathbf{b} \tag{9}$$

$$t_k \leftarrow \mathbf{A}_{k+1} \mathbf{w}_{k+1} \tag{10}$$

After all the iterations of the loop have been completed or the loop has been prematurely stopped as described previously, we form a matrix $\mathbf{W}$ whose columns are given by the vectors $\mathbf{w}_0, \mathbf{w}_1, \ldots, \mathbf{w}_{z-1}$, *i.e.* $\mathbf{W} \leftarrow \left[\mathbf{w}_0, \mathbf{w}_1, \ldots, \mathbf{w}_{z-1}\right]$.

The matrix $\mathbf{W}$ represents the latent linear projection which can also be understood as a form of supervised dimensionality reduction. The matrix $\mathbf{A}$ (*i.e.* the input feature vectors of the training dataset $\mathcal{D}$) are now projected onto this latent space greatly reducing the dimensionality to z. This projection can be written as $\mathbf{A} \leftarrow \mathbf{AW}$.

Let a multivariate gaussian distribution function g be defined as:

$$g(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^k |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})\right) \tag{11}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are parameters of the gaussian distribution, namely the mean vector and the covariance matrix. Moreover, let $s(\mathbf{A}, n)$ be the function to generate a new matrix by selecting rows from $\mathbf{A}$ that correspond to $\mathbf{b}_i = n$.

We now compute the mean vector and covariance matrix of feature vectors from each class as follows: $\mu_{y=1} \leftarrow$ mean of vectors from $s(\mathbf{A}, 1)$, $\mu_{y=0} \leftarrow$ mean of vectors from $s(\mathbf{A}, 0)$, $\Sigma_{y=1} \leftarrow$ covariance of vectors from $s(\mathbf{A}, 1)$, and $\Sigma_{y=0} \leftarrow$ covariance of vectors from $s(\mathbf{A}, 0)$.

Now the classifier function f can be written as:

$$f(\mathbf{x}, \Theta) = \frac{g(\mathbf{x}, \mu_{y=1}, \Sigma_{y=1})}{g(\mathbf{x}, \mu_{y=0}, \Sigma_{y=0})} < r \tag{12}$$

where $\Theta = \{\mu_{y=1}, \Sigma_{y=1}, \mu_{y=0}, \Sigma_{y=0}\}$ is the set of parameters that characterize f, and r is the threshold value for binary classification. Now, the classifier f has been successfully learnt from the training data $\mathcal{D}$.

The proposed classifier training process can be intuitively understood as learning a set of linear projections in a supervised way and then projecting the input feature vectors onto this linear space and then fitting a quadratic surface in this space. At test time, given an unseen feature vector $\mathbf{x}$, we use Equation 12 to compute the classification output.

3 Experiments and Results

3.1 Datasets

We use the following three publicly available datasets from the UCI Machine Learning Repository [22] to evaluate our proposed algorithm:

- `german-number-scale`: German credit data dataset where the input features correspond to attributes such as credit history, existing account status, amount of credit, number of employed years and personal status, and the output is a binary label denoting a good or bad customer. There are 24 features and 1,000 data points. The features are scaled to be within $[-1, 1]$.
- `heart-scale`: A dataset to predict the absence or presence of heart disease using features such as age, sex, type of chest pain, serum cholestoral level, fasting blood sugar level and maximum heart rate that can be achieved. The number of features is 13 and the features are scaled to be in the range $[-1, 1]$.
- `ionosphere-scale`: A dataset to classify radar returns (to "good" or "bad") collected from the ionosphere using features derived from pulse numbers of the radar system. There are 34 features and they are scaled as with the other datasets.

From each dataset, the corresponding training and test datasets are generated by randomly splitting the dataset with the 70% and 30% ratio. During the training process, the classifier is trained with the training dataset and at test time, the test dataset is used to evaluate the performance of the classifier.

3.2 Evaluation Criteria

The performance criteria used is the Receiver Operating Characteristic (ROC) curve which plots the true positive rate (TPR) versus the false positive rate

Classifier	credit	ionosphere	heart	AUC mean
`PLSQuad`	0.7846	0.9823	0.9216	0.8962
`RForest`	0.7735	0.9772	0.9309	0.8939
`RBF-SVM`	0.7687	0.9764	0.8994	0.8815
`SVM`	0.7702	0.8492	0.9117	0.8437
`Boost`	0.7245	0.9658	0.8944	0.8616

Table 1: The AUCs obtained by the classifiers on each dataset (`german-number-scale`, `ionosphere-scale` and `heart-scale` respectively).

(FPR), at varying classifier output thresholds. The TPR and FPR can be defined as follows:

$$\text{TPR} = \frac{\#\text{ true positives}}{\#\text{ false negatives} + \#\text{ true positives}} \tag{13}$$

$$\text{FPR} = 1 - \frac{\#\text{ true negatives}}{\#\text{ false positives} + \#\text{ true negatives}} \tag{14}$$

In addition to ROC curves, we also use a performance measure that summarizes an entire ROC curve which is the area under the given ROC curve (AUC). The reason for adopting the AUC and the ROC curve as the performance measures is due to the fact that they are not sensitive to class imbalance (*i.e.* the difference in the number of data in each class), and in addition, they show the overall performance of the classifier at different classifier score thresholds.

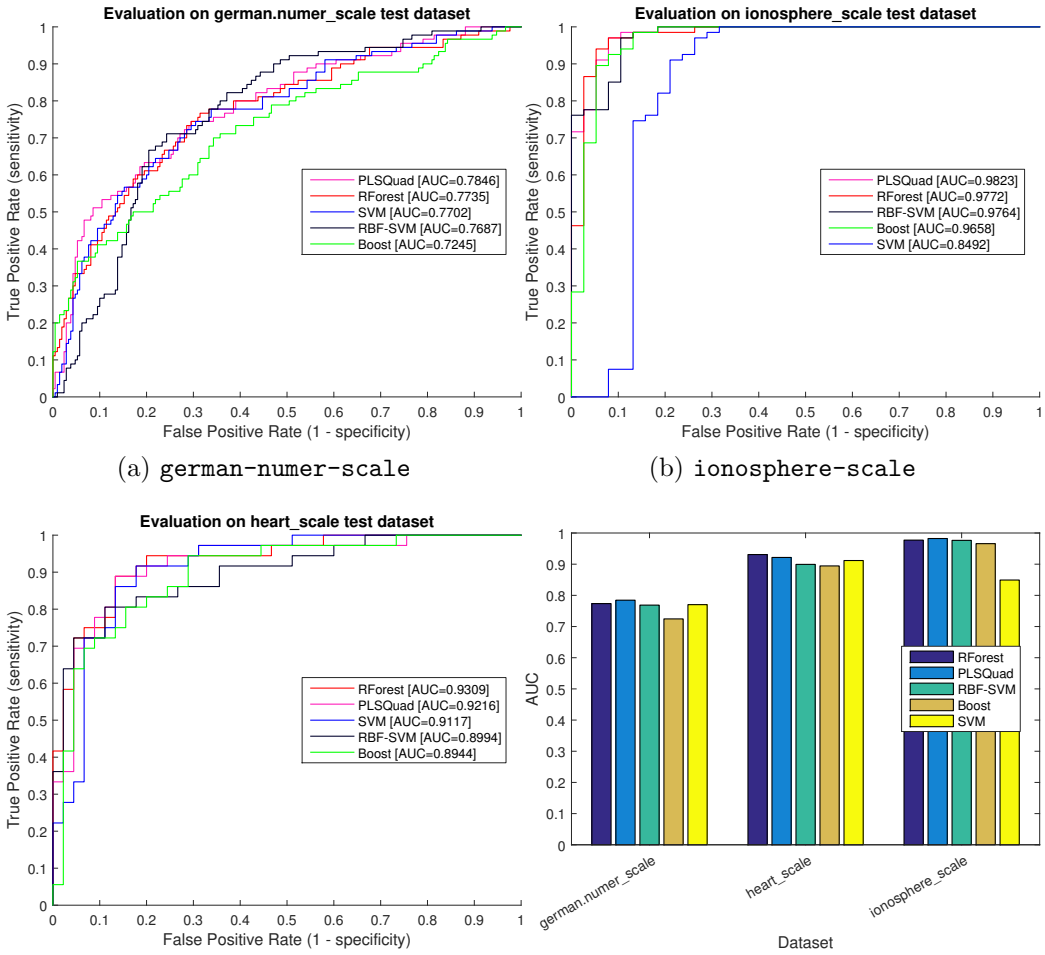
3.3 Experimental Settings

Our proposed method is compared with the state-of-the-art classification methods discussed in Section 1, namely: (1) `SVM`: Support Vector Machine with linear kernel, (2) `RBF-SVM`: Support Vector Machine with (non-linear) RBF kernel, (3) `Boost`: Real AdaBoost and (4) `RForest`: Random Forest.

3.4 Results and Discussion

The ROC curves and the corresponding AUC values are shown in Figures 1a, 1b and 1c. The AUC values and the mean AUC value across all the datasets for each classifier can found in Table 1. A bar chart visualizing the AUCs is given in Figure 1d.

As can be seen from these figures, the bar graph and the table, our proposed `PLSQuad` obtains the first place ahead of all the state-of-the-art classifiers in all the datasets except the `heart-scale` dataset where it places the second place, while having very close AUC measure (0.9216) to `RForest` which got first place (0.9309).



(a) `german-numer-scale` (b) `ionosphere-scale`

(c) `heart-scale` (d) Comparison of AUCs

Figure 1: Evaluation on the test datasets

In all the datasets, `Boost` got either last or second last place, consolidating the independent findings by other researchers that AdaBoost tends to overfit in many datasets.

Our proposed approach `PLSQuad` performs better than `RForest` in two of the three datasets and the results are significant because `RForest` has been shown to be one of the top state-of-the-art classifiers in the literature and `PLSQuad` is superior to `RForest` in the majority of the datasets, and very close to `RForest` in the other. Moreover, `PLSQuad` is significantly better than `SVM` and `Boost` is all the datasets. `PLSQuad` also has the highest mean AUC across all the datasets and therefore outperforms all the state-of-the-art classifiers on average.

4 Conclusion and Future Work

In this paper, we have proposed a novel classification algorithm called Hybrid Partial Least Squares Quadratic Discriminant that outperforms state-of-the-art classifiers such as the Random Forest, AdaBoost, linear SVM and non-linear RBF SVM. Our proposed algorithm automatically learns a set of linear projections using the supervised labels and then fits a quadratic surface in the space spanned by the basis of the linear projections. We evaluated the proposed method on three challenging datasets containing real world data and obtained promising results.

There are several potential future research directions that can make use of the method presented in this paper. Firstly, the mathematical formulation be extended to directly work with multi-class classification problems. Secondly, there is a potential to modify the proposed algorithm to learn other kinds of non-linear surfaces other than quadratic ones in the linear projection space. Thirdly, it would be interesting to apply the proposed classification algorithm to problems in various fields of Machine Learning such as computer vision, natural language processing and speech processing.

References

[1] Paul Viola and Michael J. Jones, Robust real-time face detection, *International Journal of Computer Vision*, **57** (2004), no. 2, 137–154. https://doi.org/10.1023/b:visi.0000013087.49260.fb

[2] Edgar Osuna, Robert Freund and Federico Girosit, Training support vector machines: an application to face detection, In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1997, 130–136. https://doi.org/10.1109/cvpr.1997.609310

[3] Erik Hjelmås and Boon Kee Low, Face detection: A survey, *Computer Vision and Image Understanding*, **83** (2001), no. 3, 236–274. https://doi.org/10.1006/cviu.2001.0921

[4] Biing-Hwang Juang, Wu Hou and Chin-Hui Lee, Minimum classification error rate methods for speech recognition, *IEEE Transactions on Speech and Audio Processing*, **5** (1997), no. 3, 257–265. https://doi.org/10.1109/89.568732

[5] Kyaw Kyaw Htike and David Hogg, Adapting pedestrian detectors to new domains: A comprehensive review, *Engineering Applications of Artificial Intelligence*, **50** (2016), 142–158. https://doi.org/10.1016/j.engappai.2016.01.029

[6] Kyaw Kyaw Htike, Efficient labelling of pedestrian supervisions, *ELCVIA Electronic Letters on Computer Vision and Image Analysis*, **15** (2016), no. 1, 77–99. https://doi.org/10.5565/rev/elcvia.881

[7] Kyaw Kyaw Htike and David Hogg, Efficient non-iterative domain adaptation of pedestrian detectors to video scenes, *2014 22nd International Conference on Pattern Recognition*, (2014), 654–659. https://doi.org/10.1109/icpr.2014.123

[8] Jia-Ching Wang, Jhing-Fa Wang, Kuok Wai He and Cheng-Shu Hsu, Environmental sound classification using hybrid SVM/KNN classifier and mpeg-7 audio low-level descriptor, *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, (2006), 1731–1735. https://doi.org/10.1109/ijcnn.2006.1716317

[9] Lukáš Neumann and Jiří Matas, Real-time scene text localization and recognition, *2012 IEEE Conference on Computer Vision and Pattern Recognition*, (2012) 3538–3545. https://doi.org/10.1109/cvpr.2012.6248097

[10] Kyaw Kyaw Htike and Othman O. Khalifa, Comparison of supervised and unsupervised learning classifiers for human posture recognition, *International Conference on Computer and Communication Engineering (ICCCE'10)*, (2010), 1–6. https://doi.org/10.1109/iccce.2010.5556749

[11] Kyaw Kyaw Htike, Othman O. Khalifa, Huda Adibah Mohd Ramli and Mohammad A.M. Abushariah, Human activity recognition for video surveillance using sequences of postures, *The Third International Conference on e-Technologies and Networks for Development (ICeND2014)*, (2014), 79–82. https://doi.org/10.1109/icend.2014.6991357

[12] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik, A training algorithm for optimal margin classifiers, In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, ACM, 1992, 144–152. https://doi.org/10.1145/130385.130401

[13] Corinna Cortes and Vladimir Vapnik, Support-vector networks, *Machine Learning*, **20** (1995), no. 3, 273–297. https://doi.org/10.1007/bf00994018

[14] Leo Breiman, Random forests, *Machine Learning*, **45** (2001), no. 1, 5–32. https://doi.org/10.1023/a:1010933404324

[15] Yoav Freund and Robert E. Schapire, A desicion-theoretic generalization of on-line learning and an application to boosting, Chapter in *Computational Learning Theory*, Springer, 1995, 23–37. https://doi.org/10.1007/3-540-59119-2_166

[16] Jerome Friedman, Trevor Hastie and Robert Tibshirani, Additive logistic regression: a statistical view of boosting, *Annals of Statistics*, **28** (1998), 337–407. https://doi.org/10.1214/aos/1016120463

[17] Robert E Schapire and Yoram Singer, Improved boosting algorithms using confidence-rated predictions, *Machine Learning*, **37** (1999), no. 3, 297–336. https://doi.org/10.1023/a:1007614523901

[18] Leo Breiman, Bagging predictors, *Machine Learning*, **24** (1996), no. 2, 123–140. https://doi.org/10.1007/bf00058655

[19] Tin Kam Ho, The random subspace method for constructing decision forests, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **20** (1998), no. 8, 832–844. https://doi.org/10.1109/34.709601

[20] Paul Geladi and Bruce R. Kowalski, Partial least-squares regression: a tutorial, *Analytica Chimica Acta*, **185** (1986), 1–17. https://doi.org/10.1016/0003-2670(86)80028-9

[21] Bernhard Scholkopft and Klaus-Robert Mullert, Fisher discriminant analysis with kernels, *Neural Networks for Signal Processing IX*, (1999). https://doi.org/10.1109/nnsp.1999.788121

[22] M. Lichman, UCI Machine Learning Repository, 2013.

Received: September 30, 2016; Published: November 20, 2016