Improved Adaline Networks for Robust Pattern Classification

César Lincoln C. Mattos[1], José Daniel Alencar Santos[2], and Guilherme A. Barreto[1]

[1] Federal University of Ceará (UFC), Department of Teleinformatics Engineering, Center of Technology, Campus of Pici, Fortaleza, Ceará, Brazil
cesarlincoln@terra.com.br, gbarreto@ufc.br

[2] Federal Institute of Education, Science and Technology of Ceará (IFCE) Department of Industry, Maracanaú, Ceará, Brazil
jdaniel@ifce.edu.br

Abstract. The Adaline network [1] is a classic neural architecture whose learning rule is the famous least mean squares (LMS) algorithm (a.k.a. delta rule or Widrow-Hoff rule). It has been demonstrated that the LMS algorithm is optimal in H_∞ sense since it tolerates *small* (in energy) disturbances, such as measurement noise, parameter drifting and modelling errors [2,3]. Such optimality of the LMS algorithm, however, has been demonstrated for regression-like problems only, not for pattern classification. Bearing this in mind, we firstly show that the performances of the LMS algorithm and variants of it (including the recent Kernel LMS algorithm) in pattern classification tasks deteriorates considerably in the presence of labelling errors, and then introduce robust extensions of the Adaline network that can deal efficiently with such errors. Comprehensive computer simulations show that the proposed extension consistently outperforms the original version.

Keywords: Adaptive linear classifiers, least mean squares, labelling errors, outliers, M-estimation, robust pattern recognition.

1 Introduction

Linear neural network architectures, such as the *ADA*ptive *LIN*ear *E*lement (Adaline) network [1], have been used either as a standalone device that forms itself the core of the designed intelligent system, or as a fundamental building block of more advanced multilayer nonlinear neural networks, such as the multilayer perceptron (MLP), the radial basis functions networks (RBFN) [4], the No-Propagation (No-Prop) network [5] and the echo-state network (ESN) [6].

Weights in the Adaline are updated using the well-known least mean squares[1] (LMS) algorithm, which minimizes the mean squared error (MSE) by updating the weight vector in the negative direction of the instantaneous gradient of the MSE with respect to the weight vector. It has been demonstrated that the LMS algorithm is optimal in H_∞ sense since it tolerates *small* disturbances, such as

[1] Also known as delta rule or the Widrow-Hoff rule.

S. Wermter et al. (Eds.): ICANN 2014, LNCS 8681, pp. 579–586, 2014.

measurement noise, parameter drifting and modelling errors [2,3]. However, when the disturbances are not small (e.g. presence of impulsive noise) the performance of the LMS algorithm deteriorates considerably [7].

It is important to highlight that the aforementioned studies on the robustness of the LMS algorithm have been ascertained for regression-like tasks, typically found in the signal processing domain, such as channel equalization and time series prediction. In this paper, however, we are interested in evaluating the performance of the Adaline classifier trained by means of the LMS algorithm and variants in pattern classification tasks contaminated with outliers, in particular those resulting from labelling errors[2]. It is important to emphasize that the online behavior of the Adaline/LMS algorithm is desirable in scenarios where the full dataset is not initially available.

In order to handle labelling errors efficiently, we evaluate the performance of the Adaline network as a pattern classifier for different variants of the LMS algorithm, such as the Kernel LMS (KLMS) [8] and the least mean M-estimate (LMM) [7] algorithms, in order to devise an improved robust variant for that classifier. For the sake of completeness, performance comparison of the Adaline classifier with an SVM classifier trained with the kernel Adatron algorithm [9] is also carried out.

The remainder of the paper is organized as follows. Section 2 describes all the LMS-based algorithms that will be applied. Section 3 presents the experimental results with both artificial and real data within a robust classification context. Finally, Section 4 concludes the paper.

2 The Basics of the LMS Algorithm and Variants

Given a sequence of input-output pairs $\{(\boldsymbol{x}_i, y_i)\}_i^N \in \mathbb{R}^D \times \mathbb{R}$, the corresponding output of the Adaline classifier[3] is estimated as

$$\hat{y}_i = \boldsymbol{w}_i^T \boldsymbol{x}_i, \qquad i \in \{1, \ldots, N\} \tag{1}$$

where N is the total number of available inputs, $\boldsymbol{w}_i \in \mathbb{R}^D$ is the weight vector and $\hat{y}_i \in \mathbb{R}$ is the estimated output provided by the linear model. The problem of training an Adaline classifier corresponds to the process of recursively updating the weight vector $\boldsymbol{w}_i$ for each input vector. In the following paragraphs we briefly describe the LMS and some variants of it that will be evaluated in this paper. Before proceeding, however, it is worth mentioning that the LMS variants to be described next were introduced in the context of signal processing applications and have not been evaluated before in the robust classification scenario.

[2] This type of outlier may result either from mistakes during labelling the data points (e.g. misjudgment of a specialist) or from typing errors during creation of data storage files (e.g. by striking an incorrect key on a keyboard).

[3] In this paper we discuss only binary classification problems. Thus, we need only one output neuron. Generalization of the presented concepts to multiclass problems is straightforward.

The LMS can be seen as a search algorithm in which a steepest-descent-based approach is applied to obtain a solution that minimizes the MSE:

$$J_{\text{MSE}}(\boldsymbol{w}_i) = \sum_{i=1}^{N} \mathbb{E}\{e_i^2\} = \sum_{i=1}^{N} \mathbb{E}\{(y_i - \boldsymbol{w}_i^T \boldsymbol{x}_i)^2\}, \tag{2}$$

where $\mathbb{E}\{\cdot\}$ is the expectation operator and $e_i = y_i - \boldsymbol{w}_i^T \boldsymbol{x}_i$ is the error for the i-th iteration. Minimization of the Eq. (2) is obtained by taking its gradient with respect to the weights: $\frac{\partial J_{\text{MSE}}(\boldsymbol{w}_i)}{\partial \boldsymbol{w}_i} = -2\mathbb{E}\{e_i \boldsymbol{x}_i\}$. The recursive algorithm is calculated updating $\boldsymbol{w}_i$ at each iteration in the negative direction of this gradient, which involves approximating $\mathbb{E}\{e_i \boldsymbol{x}_i\}$ by its instantaneous value $e_i \boldsymbol{x}_i$:

$$\boldsymbol{w}_{i+1} = \boldsymbol{w}_i - \mu \frac{\partial J_{\text{MSE}}(\boldsymbol{w}_i)}{\partial \boldsymbol{w}_i} = \boldsymbol{w}_i + \mu e_i \boldsymbol{x}_i, \tag{3}$$

where μ is a learning step which controls the convergence rate. The choice of the μ is problem dependent and can reduce the efficiency of the method. One possible alternative arises when a variable step size is applied. In the normalized LMS (NLMS) algorithm, the learning step is divided by the squared L_2-norm of the input as follows:

$$\boldsymbol{w}_{i+1} = \boldsymbol{w}_i + \frac{\mu}{\epsilon + \|\boldsymbol{x}\|^2} e_i \boldsymbol{x}_i, \tag{4}$$

where ϵ is a very small positive constant needed to avoid division by zero.

The second algorithm to be described is the LMM algorithm [10,7] which borrows concepts from robust statistics and the M-estimation framework introduced by Huber [11]. The goal of robust statistics is to devise parameter estimation algorithms that provides faithful estimates in modelling scenarios where the assumption of Gaussianity for estimation errors does not hold. In this regard, the LMM algorithm derives from a more general objective function than the MSE:

$$J_{\text{LMM}}(\boldsymbol{w}_i) = \sum_{i=1}^{N} \mathbb{E}\{\rho(e_i)\} = \sum_{i=1}^{N} \mathbb{E}\{\rho(y_i - \boldsymbol{w}_i^T \boldsymbol{x}_i)\}, \tag{5}$$

where $\rho(\cdot)$ is the M-estimate function [11]. The function $\rho(\cdot)$ computes the contribution of each error e_i to the objective function $J_{\text{LMM}}(\boldsymbol{w}_i)$. Note that when $\rho(u) = u^2$, the function $J_{\text{LMM}}(\boldsymbol{w}_i)$ reduces to the MSE function $J_{\text{MSE}}(\boldsymbol{w}_i)$.

Weight updating in LMM follows the same logic of the LMS algorithm:

$$\boldsymbol{w}_{i+1} = \boldsymbol{w}_i - \mu \frac{\partial J_{\text{LMM}}(\boldsymbol{w}_i)}{\partial \boldsymbol{w}_i} = \boldsymbol{w}_i + \mu q(e_i) e_i \boldsymbol{x}_i, \tag{6}$$

where $q(e_i) = \frac{1}{e_i} \frac{\partial J_{\text{LMM}}(\boldsymbol{w}_i)}{\partial \boldsymbol{w}_i}$ is the weighting function. Note that, if $\rho(e_i) = e_i^2$, then $q(e_i) = 1$, and Eq. (6) becomes equal to Eq. (3). In the present paper the modified Huber M-estimate function will be considered [7]:

$$\rho(e_i) = \begin{cases} e_i^2/2, & 0 \le |e_i| < \xi \\ \xi^2/2, & \text{otherwise} \end{cases}, \qquad q(e_i) = \begin{cases} e_i, & 0 \le |e_i| < \xi \\ 0, & \text{otherwise} \end{cases}, \tag{7}$$

where ξ is a threshold parameter which avoids the influence of inputs with large errors. Smaller values of ξ produce more resistance to outliers, but at the expense of lower efficiency when the errors are normally distributed.

If we divide the step size of the LMM algorithm by the squared L_2-norm of the input vector, we get the Normalized LMM (NLMM) algorithm [7]:

$$\boldsymbol{w}_{i+1} = \boldsymbol{w}_i + \frac{\mu q(e_i) e_i \boldsymbol{x}_i}{\epsilon + \boldsymbol{x}_i^T \boldsymbol{x}_i}, \tag{8}$$

where ϵ has the same meaning as in Eq. (4).

The third LMS-like algorithm to be described is the KLMS algorithm, which works as the LMS algorithm but now operating on the feature space obtained by applying a mapping $\Phi(\cdot)$ to the inputs, generating a new sequence of input-output pairs $\{(\Phi(\boldsymbol{x}_i), y_i)\}_{i=1}^N$ [8]. Weight updating is similar to the Eq. (3), i.e.

$$\boldsymbol{w}_{i+1} = \boldsymbol{w} + \mu e_i \Phi(\boldsymbol{x}_i). \tag{9}$$

Considering $\boldsymbol{w}_0 = \boldsymbol{0}$, where $\boldsymbol{0}$ is the null-vector, after N iterations we get

$$\boldsymbol{w}_N = \mu \sum_{i=1}^{N-1} e_i \Phi(\boldsymbol{x}_i), \qquad \hat{y}_N = \boldsymbol{w}_N^T \Phi(\boldsymbol{x}_N) = \mu \sum_{i=1}^{N-1} e_i \kappa(\boldsymbol{x}_i, \boldsymbol{x}_N), \tag{10}$$

where $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \Phi(\boldsymbol{x}_i)^T \Phi(\boldsymbol{x}_j)$ is a positive-definite kernel function. It should be noted that only Eq. (10) is needed both for training and testing. Although the values of the weight vector do not need to be computed, the *a priori* errors $e_i, i \in \{1, \cdots N\}$, and the training inputs $\boldsymbol{x}_i, i \in \{1, \cdots N\}$, must be maintained for prediction. In [12] a normalized version of the KLMS algorithm, named NKLMS, was proposed by modifying Eq. (10) as follows

$$\hat{y}_N = \mu \sum_{i=1}^{N-1} e_i \frac{\kappa(\boldsymbol{x}_i, \boldsymbol{x}_N)}{\kappa(\boldsymbol{x}_i, \boldsymbol{x}_i)}. \tag{11}$$

Since we are going to evaluate the performance of the Adaline classifier, when experimenting with KLMS and NKLMS algorithms we used their linear versions, choosing the linear kernel $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^T \boldsymbol{x}_j + C$, where C is a constant.

The last estimation algorithm to be described, the Kernel Adatron (KAdatron) [9], is an on-line algorithm for training linear perceptron-like classifiers by providing a procedure that emulates Support Vector Machines without resorting to any quadratic programming toolboxes. By writing the KAdatron algorithm in the data-dependent representation $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$, we obtain the following steps:

1. Initialize $\alpha_i = 0$ (Lagrange multipliers).
2. Calculate $z_i = \sum_{j=1}^N \alpha_j y_j \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$.
3. Calculate $\gamma_i = y_i z_i$.
4. Let $\delta\alpha_i = \mu(1 - \gamma_i)$ be the proposed change to α_i;
   - If $(\delta\alpha_i + \alpha_i) \leq 0$ then $\alpha_i = 0$.
   - If $(\delta\alpha_i + \alpha_i) > 0$ then $\alpha_i = \alpha_i + \delta\alpha_i$.

5. If a maximum number of presentations of the training set has been exceeded then stop, otherwise return to Step 2.

The estimation for a new input $\boldsymbol{x}_*$ can be written as:

$$\hat{y}_* = \sum_{i \in SV} y_i \alpha_i^o \kappa(\boldsymbol{x}_*, \boldsymbol{x}_i), \tag{12}$$

where α_i^o is the solution of KAdatron algorithm and SV represents the index set of support vectors. As with KLMS and NKLMS cases, we will also use a linear kernel for experiments with KAdatron.

3 Experimental Results and Discussion

The experimental results were separated in two groups: one with artificial 2-dimensional data for the sake of visualization of the decision regions obtained by each classifier; and other with two real-world datasets (Iris and Vertebral Column)[4], for analyzing classifiers' performance due to the presence of outliers.

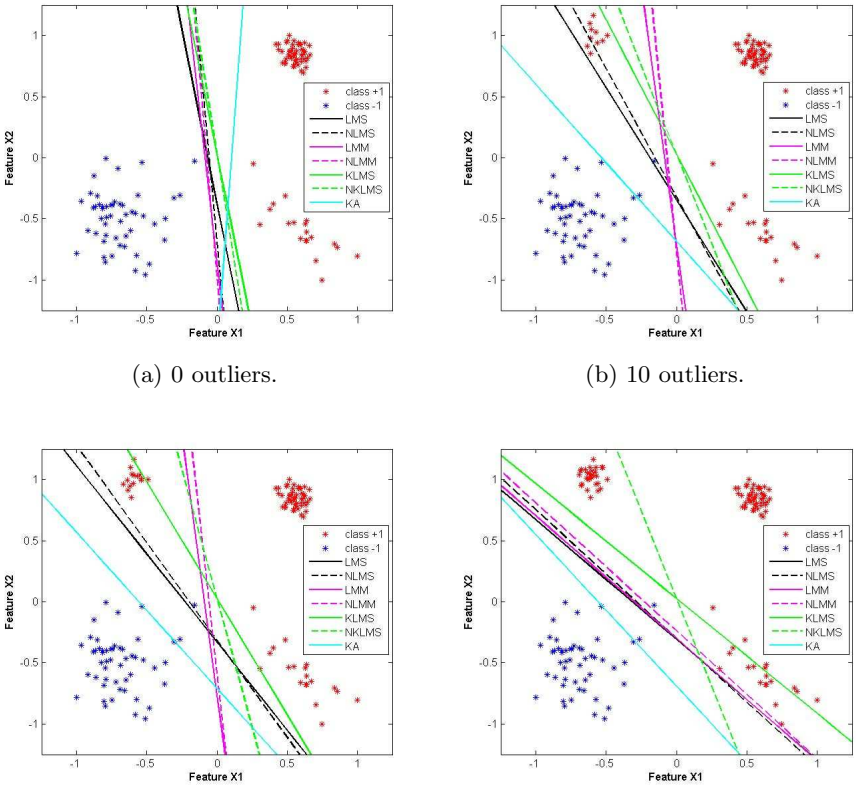
The first group of experiments involved a 120 2-dimensional samples from two classes (red and blue), which are linearly separable. All data samples are used for training the classifiers, since the goal is to visualize the final position of the decision line and not to compute recognition rates. A number of outliers from the red class was gradually added at each experiment, close to the region associated to the blue class. The obtained decision lines for each version of the Adaline classifier is shown in Figure 1. The learning step μ was set to 0.01 and the number of training epochs was set to 100 for all experiments.

One may notice that with the addition of 10 and 15 outliers, the Adaline/LMM, Adaline/NLMM and Adaline/NKLMS were less sensitive to the presence of outliers than the other classifiers. With the addition of 30 outliers, all the classifiers were strongly affected, except the Adaline/NKLMS. A remark is then required here. It is commonsense that the very nature of outliers demand that they should appear in a small number. When this number is too high, perhaps they should not be considered as outliers anymore, but as usual data samples of the class. In this situation, we recommend a more powerful classifier (e.g. the ELM or the MLP) to be used, since it can produce a nonlinear decision curve.

In the second group of experiments, the Iris dataset was prepared in the following way: the Virginica and Versicolor classes were labeled $+1$ and -1, respectively. From those two classes, 80% was used for training and 20% for testing. During training step, some samples from Setosa class were added with the label $+1$, being considered as outliers from the Virginica class. The number of generated outliers were 0%, 5%, 10%, 20% and 30% of the original number of Virginica samples in the training set.

The results obtained after 100 training-testing cycles are summarized in Table 1. It can be noticed that in the experiment without outliers, aside from Adaline/KLMS and Adaline/NKLMS, all methods achieved similar results. From

[4] Freely available at [13].



(a) 0 outliers. (b) 10 outliers.

(c) 15 outliers. (d) 30 outliers.

Fig. 1. Effect in the decision regions after adding outliers

5% to 10%, Adaline/LMM and Adaline/NLMM practically did not change their performances, while all the others methods suffered with the inclusion of outliers. With 20% of outliers, Adaline/LMM is also penalized, but Adaline/NLMM maintains high accuracy rates. With 30%, all algorithms obtain lower accuracy values. For this last case we again emphasize that those samples cannot actually be considered *outliers* anymore, but normal training samples.

The Vertebral Column dataset was configured as a 2-class problem since we removed the samples from the Disk Hernia class and considered only the ones from Normal and Spondylolisthesis classes. The same $80\% - 20\%$ partitioning was used. The addition of outliers was done by mislabeling some patterns on purpose: a portion (0%, 5%, 10%, 20% and 30%) of training samples from the Spondylolisthesis class had their labels changed to the Normal class. The results averaged after 100 repetitions are presented in Table 1.

Table 1. Results for Iris and Vertebral Column classification without and with outliers

	0%	5%	10%	20%	30%
Iris dataset					
Adaline/LMS	96.25 ± 3.72	88.95 ± 6.68	83.85 ± 7.81	75.10 ± 10.02	68.95 ± 10.88
Adaline/NLMS	96.15 ± 3.75	93.50 ± 4.58	91.90 ± 6.02	83.75 ± 8.11	77.30 ± 9.86
Adaline/LMM	95.85 ± 4.02	94.95 ± 4.17	94.90 ± 4.44	75.10 ± 9.61	69.05 ± 10.84
Adaline/NLMM	95.70 ± 3.63	94.90 ± 4.38	94.80 ± 4.76	95.10 ± 4.14	77.10 ± 9.83
Adaline/KLMS	92.15 ± 6.08	87.35 ± 8.12	85.50 ± 8.48	76.45 ± 10.08	69.15 ± 10.28
Adaline/NKLMS	91.15 ± 6.66	87.85 ± 7.73	86.50 ± 8.42	84.05 ± 8.75	79.75 ± 8.94
KAdatron	95.20 ± 5.27	86.55 ± 7.27	76.20 ± 11.17	71.60 ± 9.92	68.00 ± 9.97
Vertebral Column dataset					
Adaline/LMS	90.06 ± 4.85	89.52 ± 3.93	89.32 ± 4.32	84.22 ± 4.99	76.60 ± 7.14
Adaline/NLMS	91.10 ± 3.86	90.90 ± 3.88	90.88 ± 3.85	85.92 ± 4.41	78.52 ± 5.93
Adaline/LMM	91.90 ± 3.66	92.18 ± 3.27	92.16 ± 3.61	90.54 ± 3.61	82.02 ± 7.36
Adaline/NLMM	91.32 ± 3.63	91.36 ± 3.65	92.02 ± 3.76	88.74 ± 3.98	80.32 ± 6.32
Adaline/KLMS	85.32 ± 4.97	85.22 ± 5.17	83.78 ± 5.30	79.00 ± 5.37	68.52 ± 6.75
Adaline/NKLMS	81.02 ± 5.13	81.78 ± 5.28	83.08 ± 5.03	80.30 ± 5.77	68.24 ± 6.88
KAdatron	95.80 ± 2.66	92.98 ± 4.91	91.12 ± 5.59	85.06 ± 9.00	77.24 ± 9.55

In this case, only from 20% of outliers the Adaline/LMS, Adaline/NLMS and KAdatron algorithms suffered from performance degradation. The Adaline/NLMM classifier and mainly the Adaline/LMM classifier suffered less reduction in the mean accuracy rate when compared to the other classifiers. Similar behavior was observed for 30% of outliers. Once again the Adaline/KLMS and the Adaline/NKLMS classifiers did not achieve good overall results.

Concerning the results achieved by the Adaline/KLMS and Adaline/NKLMS classifiers when compared with Adaline/LMM, Adaline/NLMM, KAdatron and even with Adaline/LMS and Adaline/NLMS, they did not achieved good results. One possible explanation might be our choice of a linear kernel, as those methods were originally proposed mainly for non-linear applications with Gaussian and polynomial kernels [8,12]. It is also interesting to notice that although the KAdatron classifier algorithm obtained good results in the scenarios without outliers (for the Column dataset it was the best method), its results were strongly affected when for the scenarios with outliers.

4 Conclusion

In the present paper we evaluated several variants of the LMS learning rule to obtain different Adaline-like classifiers. The methods were evaluated for binary classification with outliers added during training, to check their robustness. After the experiments, the performances of Adaline/LMM and Adaline/NLMM classifiers exceeded the other techniques as the number of outliers was increased. The obtained results indicate the feasibility of the application in robust pattern recognition of algorithms usually related to filtering and regression problems. For instance, to the best of our knowledge, this is the first time the LMM, NLMM,

KLMS and NKLMS learning rules are applied to classification in the presence of outliers. Currently we are applying the concepts presented in this paper to add robustness to nonlinear classifiers, such as the Extreme Learning Machine.

Acknowledgments. The authors would like to thank FUNCAP (Fundação Cearense de Apoio ao Desenvolvimento Científico e Tecnológico) and NUTEC (Fundação Núcleo de Tecnologia Industrial do Ceará) for their financial support.

References

1. Widrow, B.: Thinking about thinking: The discovery of the LMS algorithm. IEEE Signal Processing Magazine 22(1), 100–106 (2005)
2. Hassibi, B., Sayed, A.H., Kailath, T.: H_∞ optimality of the LMS algorithm algorithm. IEEE Transactions on Signal Processing 44(2), 267–280 (1996)
3. Bolzern, P., Colaneri, P., De Nicolao, G.: H_∞-robustness of adaptive filters against measurement noise and parameter drift. Automatica 35(9), 1509–1520 (1999)
4. Poggio, T., Girosi, F.: Networks for approximation and learning. Proceedings of the IEEE 78(9), 1481–1497 (1990)
5. Widrow, B., Greenblatt, A., Kim, Y., Park, D.: The No-Prop algorithm: A new learning algorithm for multilayer neural networks. Neural Networks 37, 182–188 (2013)
6. Jaege, H.: Optimization and applications of echo state networks with leaky-integrator neurons. Neural Networks 20(3), 335–352 (2007)
7. Chan, S.C., Zhou, Y.: On the performance analysis of the least mean M-estimate and normalized least mean M-estimate algorithms with gaussian inputs and additive gaussian and contaminated gaussian noises. Journal of Signal Processing Systems 80(1), 81–103 (2010)
8. Liu, W., Pokharel, P., Principe, J.: The kernel least-mean-square algorithm. IEEE Transactions on Signal Processing 56(2), 543–554 (2008)
9. Friess, T.T., Cristianini, N., Campbell, C.: The kernel Adatron algorithm: A fast and simple learning procedure for support vector machines. In: Proceedings of the 15th International Conference of Machine Learning (ICML 1998), pp. 188–196 (1998)
10. Zou, Y., Chan, S.C., Ng, T.S.: Least mean M-estimate algorithms for robust adaptive filtering in impulsive noise. IEEE Transactions on Circuits and Systems II 47(12), 1564–1569 (2000)
11. Huber, P.J.: Robust estimation of a location parameter. Annals of Mathematical Statistics 35(1), 73–101 (1964)
12. Modaghegh, H., Khosravi, R., Manesh, S.A., Yazdi, H.S.: A new modeling algorithm-normalized kernel least mean square. In: Proceedings of the International Conference on Innovations in Information Technology (IIT 2009), pp. 120–124 (2009)
13. Bache, K., Lichman, M.: UCI machine learning repository (2014), `http://archive.ics.uci.edu/ml`