Early Stabilizing Feature Importance for TensorFlow Deep Neural Networks

Jeff Heaton, Steven McElwee, & James Cannady, Ph.D.
College of Engineering and Computing
Nova Southeastern University
Fort Lauderdale, Florida, USA

Abstract— Feature importance is the process where the individual elements of a machine learning model's feature vector are ranked on their relative importance to the accuracy of that model. Some feature ranking algorithms are specific to a single model type, such as Garson and Goh's neural network weight-based feature ranking algorithm. Other feature ranking algorithms are model agnostic, such as Brieman's perturbation feature ranking algorithm. This paper provides implementations for both the neural network weight-based and perturbation feature ranking algorithms for Google's TensorFlow Deep Learning framework. Additionally, this paper introduces a novel hybrid approach of these two feature ranking algorithms that produces a stable ranking of features earlier in the training epochs of a deep neural network. Earlier stabilization of feature rank can save considerable compute cycles during model searches and feature engineering where many representations of the feature vector must be compared. This paper demonstrates all three algorithms by empirically showing that the introduced hybrid weight perturbation algorithms achieves earlier stability than the established algorithms.

Keywords—Feature Importance, Feature Selection, Deep Neural Networks

I. Introduction

The ability to recognize and prioritize the most significant input variables to a neural network has been a focus of research for many years. The performance benefits available from effective feature selection techniques are well established, since when applied before training of and classification of a neural network, these techniques have been found to improve the learning speed, make the learner more generalizable, and simplify the representation of the data [1]. However, these techniques have yet to be realized to the fullest potential in emerging deep learning techniques.

This paper describes the development of a method to evaluate a reasonable stable set of features from a partially trained deep neural network based on TensorFlow [2]. The paper includes an overview of prior research in the area, a description of the feature importance method, and a discussion of the results of a series of evaluations conducted using TensorFlow. The primary contribution of this paper is the development of a new hybrid feature selection and ranking algorithm for deep neural networks that stabilizes earlier than established algorithms. A second contribution of this paper is the development of a toolkit for TesnorFlow that includes wrapper-based, filter-based, and embedded feature selection in addition to the new hybrid feature selection algorithm.[1]

II. Background

Feature selection algorithms can be classified under four categories: wrapper, filter, embedded, and hybrid [3]. Each of these algorithms provides different benefits for different applications and can be implemented using a variety of algorithms. This paper focuses on algorithms that are representative of each of these feature selection methods and that include an importance function, specifically: input perturbation feature importance, correlation coefficient feature importance, weight analysis feature importance, and a hybrid algorithm.

A. Input Perturbation Feature Importance

Wrapper-based feature selection methods validate the goodness of a subset of features using a learning algorithm, as opposed to the filter-based feature selection algorithms that use metrics to assess the usefulness of any single feature [4]. The process ends when an optimal ranking of features is generated. Wrapper selection methods search for possible features through the dataset using search algorithms with the subset being constantly evaluated. By using a learning algorithm for feature selection, wrapper methods have a better accuracy than filter methods. The main drawbacks of wrapper-based algorithms are their requirement for significant computational resources, in addition to potential for overfitting the learning algorithm [5].

Wrapper algorithms, such as the input perturbation feature importance algorithm, make use of both the neural network and a dataset. It is not necessary that the dataset used for this algorithm be the same as the one with which the neural network that it was trained.

The input perturbation feature importance algorithm calculates the loss of a neural network when each of the input features to the neural network is perturbed by the algorithm. The idea is that when an important input is perturbed the neural network should have a loss increase that corresponds to the importance of that input. Because the inputs are being perturbed, rather than removed entirely, it is not necessary to train a new neural network for each evaluated feature. Rather, the feature is perturbed in the provided dataset. The feature is

[1] Toolkit available from: https://github.com/drcannady

perturbed in such a way that it provides little or no value to the neural network, yet the neural network retains an input neuron for that feature. No change is made to the neural network as each input is evaluated.

To successfully perturb a feature for the input perturbation feature importance algorithm two objectives must be met. First, the input feature must be perturbed to the point that it now provides little or no predictive power to the neural network. Secondly, the input feature must be perturbed in such a way so that it does not have adverse effects on the neural network beyond the feature being perturbed. Both objectives are accomplished by shuffling, or perturbing, the column that is to be evaluated. By shuffling the column, the wrong input values will be presented for each of the expected targets. Secondly, the shuffle ensures that most statistical measures of the column remain the same, as the column will maintain the same distribution.

B. *Correlation Coefficient Feature Importance*

Filter-based feature selection algorithms use metrics to classify each feature. Low ranking features are eliminated [6]. The intrinsic characteristics of the data are considered in evaluating the fitness of the feature subset. Filter-based feature selection techniques do not use a learning algorithm and require fewer computing resources; however, the resulting subset of features may not be a good match for the classification algorithm [7].

Correlation coefficient feature importance is a filter algorithm that calculates the absolute value of the correlation coefficient between each of a neural network's input features. This value can be used to estimate the importance of each input feature to the neural network or any other model. The higher the correlation coefficient, the greater a feature's importance. A vector that contains the absolute value each input feature is usually normalized so that the entire vector sums to 1. This is accomplished by dividing the absolute value of each coefficient by the summation of all of these absolute values.

C. *Weight Analysis Feature Importance*

Embedded feature selection occurs naturally and without an extra filtering step as part of the training process for certain learning algorithms, such as decision trees and neural networks [8].

A number of embedded algorithms have been implemented that derive feature importance from the weights of a neural network. Many of these weight analysis algorithms, notably the Garson algorithm and the weight pairs algorithm, require a single hidden layer and are not compatible with deep neural networks. Because of this, the weight analysis based feature importance algorithm implemented by this toolkit is a simplification of algorithms proposed by the Garson and connection weights algorithms. Existing weight-based algorithms consider both the weights between the input and hidden layer, as well as the corresponding weights between the hidden layer and the output layer. When only a single hidden layer is present, it is important that the weights between both layers be considered. Figure 1 shows which weights are considered when the importance of a given neuron is calculated by either the Garson or connection weights algorithms. In this figure the importance of input I1 is being considered. The solid lines represent the weights that are used by the algorithms and dashed lines are not important for the calculation. Hidden neurons are labeled H, inputs are labeled I, bias are labeled B and the output is labeled O.

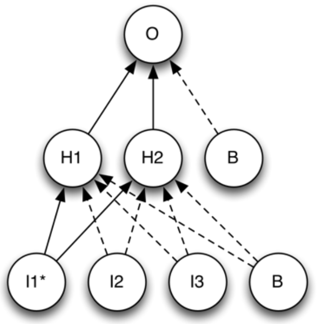

Fig. 1. Neural network weights used for importance calculation

The weight between the input and each of the hidden neurons is the primary indicator of that input's importance; however, the Garson and connection weights algorithms achieve greater accuracy by multiplying the input to hidden weights by the corresponding hidden to output weight. This allows their importance to be scaled by the output weights that they will ultimately be multiplied against. Neither algorithm makes use of the bias weights in the neural networks.

D. *Hybrid Feature Improtance Approaches*

Hybrid feature selection methods evaluate the validity of subsets from the original dataset using an independent measure in conjunction with a learning algorithm [9]. There are many examples of this method in the literature. In [10] a hybrid-feature selection based on ant-colony optimization and mutual information (ACOMI) for forecasters at the Australian Bureau of Meteorology was developed. Improved performance compared to ant-colony optimization or mutual information used separately was produced. In [11] a feature-selection algorithm based on a mixture of particle swarm optimization method and mutual information (PSO-MI) was proposed. PSO-MI showed an improved accuracy on a dataset of transient myoelectric signal, compared to particle swarm optimization or mutual information used separately. Zhang, et al, developed a two-stage selection algorithm by combining Relief and minimum-Redundancy Maximum-Relevancy (mRMR) for gene expression data [12]. The authors performed experiments comparing the mRMR-ReliefF selection algorithm with ReliefF, mRMR, and other feature selection methods using Support Vector Machine (SVM) and Naive Bayes. Experiments showed improved results using mRMR-ReliefF algorithm for gene selection compared to that of mRMR or ReliefF used separately.

In [13] a hybrid evolutionary algorithm based on the combination of ant colony optimization and simulated annealing (ACO-SA) was proposed. The initial choice of cluster center was chosen with the help of ACO and SA in order to achieve global optima. Ant colony optimization is used to find colonies between data points. Simulated annealing is used as a good local search algorithm for finding the best global position using the cumulative probability. [14] describes a hybrid algorithm combining the cAnt-Miner2 and the mRMR feature selection algorithms. cAnt-Miner2 algorithm [15] is an extended approach of coping with continuous attributes introduced by the cAnt-Miner algorithm. cAnt-Miner algorithm [16] is an extension of Ant-Miner [17]

Ant-Miner functions with continuous attributes and therefore incorporates an entropy-based discretization method during the rule construction process. cAnt-Miner creates discrete intervals for continuous attributes on the fly and does not require a discretization method for preprocessing. Experimental results of the combination of cAnt-Miner2 and mRMR using public medical data sets yielded improved results compared to that of cAnt-Miner2 only. The proposed combination was better in terms of accuracy, simplicity, and computational cost compared to the original cAnt-Miner2 algorithm.

In [18] a support vector recursive feature elimination approach was applied for gene selection incorporating an mRMR filter. The approach improved the identification of cancer tissue from benign tissues on several benchmark datasets because it accounted for the redundancy among the genes compared to mRMR or SVM-RFE separately. [19] described an approach for hyper-spectral data dimensionality reduction based on a measure of mutual information (MI) and principal components analysis (PCA) called MI-PCA, using a mutual information measure to find principal components which are spatially most like all the target classes. The authors conducted experiments using hyper-spectral data with 191 bands covering the Washington, DC area; results showed that two features selected from 191 using MI-PCA provided 98% and 93% classification accuracy for training and test data respectively, with a support vector machines classifier.

Other work on feature selection has also evaluated the ability of several evolutionary algorithms including binary particle swarm optimization and different evolution in selecting the best feature set [20]. Each of the evolutionary algorithms were then applied using both a neural network and a support vector machine to see which algorithm consistently produced the best and most concise set of features. Their experiments showed that the different evolution algorithm routinely achieved a higher prediction accuracy using a smaller number of features. Using different evolution, the total number of features selected was cut by 68%, providing a substantial reduction in the overall state space which must be maintained by the machine learning algorithm used for classification.

While numerous techniques have been applied to improve feature selection, the relative importance of individual features to the analysis performed by a neural network is the problem that is the focus of this research. Garson [21] created an algorithm that could rank the importance of the input neurons in order to reveal the behavior of neural networks by summing all of the weights between the input being evaluated and the output. Goh [22] used backpropagation neural networks to determine the importance of certain variables in geotechnical applications through an examination of the connection weights. Goh's approach provided additional detail and empirical validation of Garson's algorithm.

Olden [23] extended Goh's approach and used Garson's algorithm, and others, to better understand the mechanics of a neural network. They developed a randomization approach that was able to access the statistical importance of connection weights and the contribution of individual input variables in the neural network. The approach enabled them to interpret the individual and interacting contributions of the various input variables to the neural network. More importantly, the randomization approach employed facilitated the identification of the variables that contributed most importantly to the predictions produced by the neural network. This allowed null-connections of neurons that did not significantly contribute to the analysis to be eliminated. Olden's approach is commonly referred to as the connection weights algorithm. The algorithms proposed by Garson, Goh and Olden require a single hidden layer and are not directly compatible with deep neural networks.

While all neural network implementations can benefit from the recognition of key features in the input space deep learning-based neural networks gain particular advantages from the use of effective feature selection/importance techniques. Among deep generative models TensorFlow [2] is being applied to a variety of increasingly complex data sets. The potential offered by TensorFlow can be significantly increased if the training time required for applications using the package are reduced. However, prior to this research there was no way to evaluate a reasonable stable set of features from a partially trained deep neural network based on TensorFlow.

III. Approach

This study proposes a novel hybrid feature ranking algorithm for deep neural networks. This algorithm has the advantage of quickly converging to a stable ranking of feature importance. This requires fewer training iterations when assessing the importance of many features. In addition, this study developed a toolkit for TensorFlow that implements the feature ranking algorithms described above: input perturbation feature importance, correlation coefficient feature importance, weight analysis feature importance, and the new hybrid algorithm.

A. TensorFlow Toolkit

A toolkit was created for TensorFlow that implements each of the ranking algorithms. The TensorFlow based feature importance toolkit was implemented in the Python programming language, using the Numpy numerical framework.

The toolkit implements the individual feature importance algorithms as the following Python classes:

- CorrelationCoefficientRank
- InputPerturbationRank
- WeightRank
- HybridRank

A ranking function is provided for each in the following structure:

```
def rank(self, x, y, network):
```

The *x* and *y* parameters specify the input and target values. The *network* parameter specifies the neural network to rank. Not all ranking algorithms require all parameters to be defined.

Sample data from the University of California Irvine Machine Learning Repository was used to evaluate the performance of all four algorithms. A neural network with hidden layer counts of 200, 100, 50, and 25 neurons was used. This neural network used the ReLU [24] transfer function for hidden layers and a linear transfer function for the output layer. The Adaptive Moment Estimation (ADAM) [25] training algorithm was used with a learning rate of 0.01, a mini-batch size of 32, and all other training parameters as suggested by the original authors. A validation set of 25% was set aside with the remaining 75% used for training. The neural networks were trained until the validation set did not improve for 200 iterations.

B. Correlation Coefficient Feature Importance

The correlation coefficient feature importance provided by this toolkit is demonstrated by the following pseudocode:

```
function rank_stat(self, x, y):
  impt = []

  for i in range( numcols(x) ):
    c = corrcoef(x[: , i], y[: , 0])
    impt[i] = c

  impt = impt / sum(impt)
  return (impt)
```

This function loops over all of the input features (*x*) and calculates the correlation coefficients (*c*) between each and the target (*y*).

This simple feature importance technique requires no trained model. The feature importance provided by correlation coefficient feature importance is entirely univariate in the sense that the importance of features is considered independently. Multivariate feature analysis recognizes the fact that often one feature will influence the importance of another feature. For complex datasets, with many interactions between the input features, the univariate nature of correlation coefficient feature importance can be a limitation. Because correlation coefficient feature importance is dependent entirely on the dataset, and does not consider the model, feature importance remains the same for each training iteration of a neural network. In this sense the measure is stable; however, the univariate nature of this approach decreases its accuracy for complex datasets.

C. Weight Analysis Feature Importance

Because the toolkit implemented for this paper is designed to work with TensorFlow, no algorithm that requires a single hidden layer was implemented. Rather, the weight analysis algorithm provided by the toolkit only makes use of the input to first hidden layer hidden weights. The toolkit's weight analysis algorithm is implemented in this manor for two reasons. First, the original papers for the Garson and connection weights algorithms did not provide any direction for implementation of their algorithms beyond a single hidden layer.

Secondly, as many layers and connections are added, the direct effect of each hidden weight on the importance measure of each input neuron's weight is mitigated. The toolkit algorithm, using only the input weights, provided reasonably close assessments of the feature importance compared with the perturbation algorithm, as demonstrated by the following algorithm:

```
function rank_weight(x, y, network):
  weights = network.get_tensor_value(
    'dnn/layer0/Linear/Matrix:0')
  weights = weights ^ 2
  weights = sum(weights, axis=1)
  weights = sqrt(weights)
  weights = weights / sum(weights)
  return weights
```

First the weight matrix is retrieved from TensorFlow as a matrix of values between the input and first hidden layers. These weights are squared to preserve only the magnitude. The weights are summed along the rows to produce a vector equal to the number of input neurons. Bias values are not retrieved from TensorFlow. The square root of this vector is calculated to return the values in the same unit as the original weights. The resulting vector provides an indication of the importance of the input neurons. To normalize this vector to sum to 1, the vector is divided by the sum of the elements of the vector.

D. Input Perturbation Feature Importance

The toolkit provides an input perturbation feature importance algorithm based on the following pseudocode:

```
function rank_perturb(x, y, network):
  impt = dim(x.shape[1])

  for i in range(numcols(x)):
    hold = copy(x[, i])
    shuffle(x[, i])
    pred = network.predict(x)
    mse = mean_squared_error(y, pred)
    impt[i] = mse
    x[:, i] = hold
impt = impt / sum(impt)
return impt
```

The input perturbation algorithm begins by looping over all of the inputs in the dataset. These inputs are represented by the columns of the *x* matrix. A copy of each column is stored in the *hold* variable. Next, the contents of the column are shuffled. Predictions are generated for the neural network, and the mean square error is kept for this shuffled column. A vector of the mean square errors for each column is stored into the *impt* vector. These values are normalized to sum to 1 by dividing the *impt* vector by the sum of the elements of the *impt* vector.

E. Early Stabilizing Hybrid Feature Importance

The final algorithm provided by the toolkit is the hybrid early stabilizing algorithm introduced by this paper. This algorithm has the advantage of quickly converging to a stable ranking of feature importance. This requires fewer training iterations when assessing the importance of a large number of features.

This algorithm calculates a dynamically updated coefficient that is applied against the results of the correlation coefficient, weights, and perturbation algorithms:

$$d = \mathrm{SD}\left(\frac{p}{\sum p}\right) \tag{1}$$

This coefficient (d) is calculated by taking the standard deviation of the perturbation ranking algorithm's normalized output vector (p). The reasoning behind this equation is that the perturbation algorithm will typically begin with all features at essentially the same ranking, thus yielding a standard deviation close to zero. This is normalized by the sum of the vector all perturbation mean square errors. If this vector has already been normalized, the denominator is unneeded, as it will sum to one.

The coefficient (d) becomes the weighting between the perturbation ranking algorithm (p) and the correlation coefficient (s). This vectored weighted sum, the hybrid importance (m), is calculated by adding the weight vector, d applied to the perturbation vector (p), and the correlation coefficient (s):

$$m = w + (p \cdot d) + (s(1 - d)) \tag{2}$$

The vector m should be normalized to sum to 1, just as was done for the previous weight ranking algorithms, by dividing the vector by its summation. The hybrid algorithm is implemented as follows:

```
function rank_hybrid(x, y, network):

  p_rank = rank_perturb(x, y, network)
  w_rank = rank_weight(network)
  s_rank = rank_stat(x, y)

  d = (np.std(p_rank / sum(p_rank)))

  impt = w_rank + (p_rank * d)
         + (s_rank * (1.0 - d))

  impt = impt / sum(impt)
```

The hybrid algorithm first calculates the values of p, w and s from their respective feature ranking algorithms. The d coefficient is calculated based on the perturbation algorithm. The final importance vector is calculated according to Equation 2.

IV. Results

The results of this paper examine both the stability of the hybrid feature ranking algorithm and its ability to converge to a final rank that is similar to the perturbation algorithm. The perturbation algorithm was used as the baseline to compare against because it was one of the top algorithms evaluated by [26] that is compatible with deep neural networks.

TABLE I. Auto MPG Feature Ranking for Perturbation Ranking Algorithm

step	diff	1	2	3	4	5	6	7	8	9
0	0.13	yr	wt	dp	hp	o3	o2	ac	o1	cl
62	0.11	wt	yr	dp	hp	cl	o3	o1	ac	o2
124	0.09	wt	yr	dp	hp	o3	ac	o2	o1	cl
186	0.10	wt	yr	dp	hp	o1	ac	o3	o2	cl
248	0.09	wt	yr	dp	hp	cl	ac	o3	o2	o1
310	0.06	wt	yr	hp	dp	cl	o1	o3	o2	ac
372	0.05	wt	hp	yr	dp	o3	o1	ac	o2	cl
434	0.04	wt	hp	dp	yr	cl	o3	o1	ac	o2
496	0.02	wt	hp	yr	dp	cl	o1	ac	o3	o2
558	0.04	wt	hp	yr	dp	o1	o2	ac	o3	Cl
589	0.02	wt	hp	yr	dp	cl	o1	o3	ac	o2

TABLE II. Auto MPG Feature Ranking for Hybrid vs Perturbation Algorithm

steps	diff	1	2	3	4	5	6	7	8	9
0	0.07	wt	dp	hp	cl	yr	o1	o3	ac	o2
50	0.06	wt	dp	hp	cl	yr	o1	o3	ac	o2
100	0.06	wt	dp	hp	cl	yr	o1	o3	ac	o2
150	0.06	wt	dp	hp	cl	yr	o1	o3	ac	o2
200	0.05	wt	dp	hp	cl	yr	o1	o3	ac	o2
250	0.04	wt	dp	hp	cl	yr	o1	o3	ac	o2
300	0.04	wt	dp	hp	cl	yr	o1	o3	ac	o2
350	0.04	wt	dp	hp	cl	yr	o1	o3	ac	o2
400	0.02	wt	dp	hp	cl	yr	o1	o3	ac	o2
450	0.02	wt	dp	hp	cl	yr	o1	o3	ac	o2
475	0.03	wt	dp	hp	cl	yr	o1	o3	ac	o2

The following data sets were used from the UCI machine learning repository:

- Auto MPG Data Set
- Liver Disorders Data Set
- Breast Cancer Wisconsin (Diagnostic) Data Set

To understand what is meant by quick stabilization of a dataset, consider Table I, which shows the importance rank of the 9 features of the Auto MPG Data Set at a number of steps of training. The numbered columns show the rank of each feature, where 1 is the most important and 9 is the least. For brevity, the feature names are abbreviated; for example *yr* is year and *wt* is weight. The *diff* column shows the RMSE between the current feature ranking and what it will ultimately become. The lower the *diff* column, the closer that the current raking has converged to the final ranking. As the table demonstrates, the feature rank changes considerably as training progresses. Because the *diff* column compares to the ultimate ranking it was necessary to calculate this column at the end of processing. The *diff* column effectively indicates how similar the ranking of a particular row is to the final ranking, thus gaging how quickly an algorithm converges. The similarity between the current row and the final row is the RMSE measure between the importance vector of the importance algorithm being evaluated for the current step and the final importance vector.

Table II shows how the hybrid algorithm compares with the perturbation algorithm. The *steps* column indicates how many training steps were executed for both the perturbation and hybrid algorithms. The *diff* column indicates how close (RMSE) the hybrid algorithm is to the final baseline perturbation ranking vector. The remaining columns indicate the feature importance ranking order for the hybrid algorithm. As can clearly be seen, the ranking stabilizes quickly. The hybrid ranking algorithm converges to about 0.06 similarity to the perturbation algorithm and remains there. At step 50 the hybrid algorithm is at 0.06; whereas, the perturbation algorithm is still at 0.13.

A summary of the hybrid algorithm's performance is given in Table III for additional datasets. This table demonstrates the number of steps the hybrid algorithm required to converge compared with the number of steps needed for the perturbation algorithm. The RMSE difference between the converged perturbation ranking and the hybrid and correlation algorithms are also given.

Table 3 shows that the hybrid algorithm required fewer steps to converge than the perturbation algorithm. Additionally, even with these fewer steps, the hybrid algorithm had a relatively small difference to the converged perturbation vector. It is also important to compare the hybrid difference to the correlation coefficient feature importance. The correlation coefficient feature importance requires no training at all, but does not capture interactions between the features. For simple data sets, that do not have many interactions among features, the simple correlation feature importance algorithm is more than sufficient. This is clearly the case for the Wisconsin breast cancer data set. The difference between the correlation algorithm and hybrid was the same. However, for the more complex liver disorders data set, the hybrid algorithm was able to give a much more accurate feature rank with only a few training steps.

V. Conclusion

This paper presents a novel hybrid feature importance algorithm for deep neural networks that provides early stabilization and comparable feature importance to other established feature ranking algorithms. Effective feature ranking improves the computational performance and generalizability of the deep neural network, and earlier stabilization reduces the computational demands of feature selection. The testing results demonstrate that this novel hybrid algorithm requires fewer steps to converge than the perturbation algorithm. In addition, this paper contributes a toolkit for feature ranking in Google's TensorFlow that includes methods for input perturbation, correlation coefficient ranking, weight analysis, and the new hybrid algorithm.

TABLE III. Multiple Data Set Algorithm Comparison

Data Set	Hybrid Steps	Perturbation Steps	Hybrid Diff	Correlation Diff
Auto MPG	25	475	0.05	0.09
Liver	6	114	0.08	0.23
WC Breast	13	247	0.08	0.08

References

[1] L. C. Molina, L. Belanche, & À. Nebot. "Feature selection algorithms: a survey and experimental evaluation". IEEE International Conference on Data Mining, 2002. 306-313.

[2] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, GS. Corrado, A. Davis, J. Dean, M. Devin & S. Ghemawat. "TensorFlow: Large-scale machine learning on heterogeneous systems", 2015. Software available from tensorflow. org. 2015;1.

[3] H. Lui & L. Yu. "Toward integrating feature selection algorithms for classification and clustering". IEEE Transactions on Knowledge and Data Engineering, 17(4), 2005. 491-502.

[4] I. Guyon & A. Elisseeff. "An introduction to variable and feature selection". Journal of Machine Learning Research, 3, 1, 2003. 157-1,182.

[5] R. Kohavi & D. Sommerfield. "Feature subset selection using the wrapper model: Over fitting and dynamic search space topology". Proceedings of the International Conference on Knowledge Discovery and Data Mining, 1, 1995. 192-197.

[6] F. Ahmad, N. Norwawi, S. Deris & N. Othman. "A review of feature selection techniques via gene expression profiles". Proceedings of the International Symposium on Information Technology, 2, 2008. 1-7.

[7] H. Zou, T. Hastie & R. Tibshirani. « Sparse principal component analysis". Journal of Computational & Graphical Statistics, 15(2), 2006. 265-286.

[8] Q. Song, J. Ni, & G. Wang. "A fast clustering-based feature subset selection algorithm for high-dimensional data". IEEE Transactions on Knowledge and Data Engineering, 25(1), 2013. 1-14.

[9] S. Das. "Filters, wrappers and boosting-based hybrid for feature selection". Proceedings of the International Conference on Machine Learning, 18, 2001. 74-81

[10] C. Zhang & H. Hu. "Feature selection using the hybrid of ant colony optimization and mutual information for the forecaster". Proceedings of the International Conference on Machine Learning and Cybernetics, 3, 2005. 1728-1,732.

[11] R. Khushaba, A. Al-Ani & A. Al-Jumaily. "Swarm intelligence based dimensionality reduction for myoelectric control". Proceedings of the International Conference on Intelligent Sensors, Sensor Networks and Information, 3, 2007. 577-582.

[12] Y. Zhang, C. Ding & T. Li. "A two-stage gene selection algorithm by combining reliefF and mRMR". Proceedings of the IEEE International Conference on Bioinformatics and Bioengineering, 7, 2007. 164-171.

[13] B. Firouzi, T. Niknam, & M. Nayeripour. "A new evolutionary algorithm for cluster analysis". Proceedings of World Academy of Science, Engineering and Technology, 36, 2008. 584-588.

[14] I. Michelakos, E. Papageorgiou & M. Vasilakopoulos. "Hybrid classification algorithm evaluated on medical data". Proceedings of the IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises, 2010. 98-103.

[15] I. Michelakos, E. Papageorgiou & M. Vasilakopoulos. "A Study of cAnt-Miner2 parameters using medical data sets". Proceedings of the IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises, 19, 2010. 119-121.

[16] F. Otero, A. Freitas & C. Johnson. “cAnt-Miner: An ant colony classification algorithm to cope with continuous attributes”. Proceedings of the International Conference on Ant Colony Optimization and Swarm Intelligence, 6, 2008. 48-59.

[17] R. Parpinelli, H.Lopes & A. Freitas. “Ant-Miner is an ant colony optimization algorithm for classification task”. Proceedings of the IEEE Transactions on Evolutionary Computation, *6*(4), 2002. 321-332.

[18] P. Mundra & J. Rajapaske. “SVM-RFE with MRMR filter for gene selection“. Proceedings of the IEEE Transactions on NanoBioscience, 9(1), 2010. 31-37.

[19] M. Hossain, M. Pickering & X. Jia. “Unsupervised feature reduction based on a mutual information measure for hyperspectral image classification”. Proceedings of the Australian Space Science Conference, *11*, 2011. 1720-1723.

[20] S. Zaman, M. El-Abed & F. Karray. “Features selection approaches for intrusion detection systems based on evolution algorithms”. Proceedings of the 7th International Conference on Ubiquitous Information Management and Communication, Kota Kinabalu, Malaysia 2013.

[21] G. Garson. “Interpreting neural-network connection weights”. Artificial Intelligence Expert 6, 1991. 47–51.

[22] A. Goh. “Backpropagation neural networks for modeling complex systems”. Artificial Intelligence in Engineering, Volume 9 (3). 1995. 143-151.

[23] J. Olden & D. Jackson. “Illuminating the ‘‘black box’’: a randomization approach for understanding variable contributions in artificial neural networks”. Ecological Modeling 154, 2002. 135-150.

[24] R Hahnloser, R. Sarpeshkar, M. Mahowald, R. Douglas, & H. Seung. “Digital selection and analogue amplification coesist in a cortex-inspired silicon circuit”. Nature. 405, 2000. pp. 947–951.

[25] D. Kingma, & J. Ba. “Adam: A method for stochastic optimization.” arXiv preprint arXiv:1412.6980, 2014.

[26] J. Olden, M. Joy, & R. Death. "An accurate comparison of methods for quantifying variable importance in artificial neural networks using simulated data." Ecological Modelling 178.3, 2004. pp 389-397.