Using Random Probes for Neural Networks Based Features Selection

Hazem Migdady
Department of Computer Science
Southern Illinois University, Carbondale IL

Norman Carver
Department of Computer Science
Southern Illinois University, Carbondale IL

Abstract- *Feed forward artificial Neural Networks with backpropagation learning algorithm are of the efficient classification and pattern recognition tools that are robust to noise and can learn the target function of many learning tasks. Even though it still suffer of the long training time which limits its efficiency especially over online tasks and high dimensional datasets it is still desirable. In the context of improving neural network efficiency, it is useful to apply features selection principles that can reduce the number of neural network inputs which in turn reduces the required training time and enhances its generalization capability to end up with a neural network based classifier that perfectly matches the selected features set with good classification accuracy.*

Keywords: Neural Networks, Features Selection, Classification, Machine Learning, Random Probes.

1. Introduction

FEED forward artificial Neural Networks with the backpropagation learning algorithm are efficient tools in learning tasks that involve classification and pattern recognition. Neural networks are robust to noise and have the ability to converge to the target function to be learned by approximating the values of the desired target functions. [1] In his definition for neural networks, Negnevitsky believes that:

"[Neural Network] *is a model of reasoning based on the human brain. Thus, a neural network is considered as a highly complex, nonlinear, and parallel information processing system*" [2]

Even though multilayer feed forward neural networks with backpropagation learning algorithm are computationally expensive due to long training times, they are still desirable since they can converge over many learning tasks. Thus, many efforts have been done to improve backpropagation neural networks performance and increasing its efficiency, especially its generalization and classification ability.

A critical factor that affects neural networks performance is the number of its inputs. Experiments from previous works have shown that an abundance of a neural network inputs (*i.e.* features) often results in overfitting and poor generalization for the classifier, while if less inputs than necessary are used this limits the efficiency and capability to converge to the target function. Features can be categorized as: 1) relevant features and 2) irrelevant features which may be contained in any dataset.

Relevant features are those affect the underlying structure of the data and provide enough information about the target, while irrelevant features do not. [3]

In reference [4] the authors defined relevant and irrelevant features using the conditional probability. Under the assumption that the feature values are discrete, F is a random variable of features $[F_1, F_2 \dots F_n]$ then a pattern vector $f = [f_1, f_2 \dots f_n]$ is a realization of F. Hence feature F_i is surely irrelevant iff for all subset of features K^{-i} including F^{-i}:

$$P(F_i, Y \middle| K^{-i}) = P(F_i \middle| K^{-i}) P(Y \middle| K^{-i})$$

F^{-i} is a subset of F excluding feature f_i, K^{-i} is a subset of F^{-i} and Y is the target which is a random variable taking values y. This definition implies that feature F_i does not affect the value of the target variable Y (*i.e.* Y is independent of F_i), hence F_i is irrelevant to Y.

In this paper, we introduce a novel method that applies features selection concepts and sieves the original candidate features in a dataset to explore its intrinsic dimensionality and to remove irrelevant features. This improves neural network performance by recognizing and keeping relevant features, which enhances the generalization capability of the classifier and saves resources in any future data gathering.

2. Related Work

A number of approaches and techniques have been proposed to overcome the curse of high dimensional datasets (*i.e.* dataset with a large amount of features) that limits the efficiency of a multilayer feed forward neural network. A dataset with a large number of features implies that the neural network receives a large number of inputs, which in turn increases the required training time and computations, especially in fully connected networks.

Features selection methods can be categorized mainly onto two categories: 1) Filters and 2) Wrappers. Filters are techniques that select features without taking into account the optimization of a learning machine topology and its performance (*i.e.* preprocessing, predictor independent step –

model free techniques). On the other hand, wrappers involve the process of predictor optimization as a correlated step to the features selection process. Thus, even though they are computationally more intensive, they have the advantage of avoiding the problem that the selected features subset with filters may not match the selected predictor perfectly, which may result in poor performance of a classifier. In the case of wrappers, a learning machine performance is utilized to evaluate features subsets according to their predictive power. [4]

In their approach, Heuristic for Variable Selection (HVS), Yacoub and Bennani [5] suggested a feed forward neural network based wrapper that tries to reduce the number of input features according to network weights behavior during training process. At each training session, the input feature with the lowest weights will be pruned. Even though HVS is simple and easy in the sense that it does not involve complicated calculus and it has good results in comparison with other methods, it requires a number of training sessions equal to the number of irrelevant and redundant features to be pruned, which might be large. This can limit HVS performance by increasing the required search time, taking into account that the search depends on the classifier performance.

Another method described in [3] contains two phases, a filter phase followed by a wrapper phase. The filter phase sieves features using a genetic algorithms technique. The second phase starts as a wrapper by presenting the selected features from the first phase as inputs to a feed forward neural network, in order to recognize and remove redundant features according to that network's performance. This method removes features by filtering without taking into account the performance of the produced classifier, since the fitness function in the genetic algorithm evaluates features according to cost and inconsistency measures which are not critically related to the classifier performance. Moreover this method consumes a large amount of neural network training even though it is not necessary to retrain the network after each reduction. The authors tried to overcome the problem of training the neural network over the entire set of features by using genetic algorithms, but genetic algorithms also involve a large amount of computation.

It is possible to note that most of the methods which aim to improve neural network performance are based on the weights behavior during the training process of the network. The main limitation for a neural network is its training cost over the entire features in a dataset especially over those with high dimension. Thus, some methods try to reduce the number of features in a separate step before presenting data to the neural network, as in [3]. Even though such an approach saves time, it has risks removing some features that may have a critical effect of improving classifier performance in combination with other features.

Another method that follows the same strategy was proposed in [6]. This method is a filter method that starts by ranking all features using Gram-Schmidt orthogonalization technique. [7] After that, a threshold is ranked and inserted in that list. This threshold acts as a boundary between relevant and irrelevant features. All features that are ranked lower than that threshold will be discarded since they are considered irrelevant features. After that, the selected features are presented to a fully connected feed forward neural network to be trained. In this approach the classifier is not involved in the features selection process which reflects on its performance since the selected features are not necessarily will match the classifier perfectly even though the classifier will converge over them.

An approach uses the sensitivity analysis for features selection was explained in [8]. The main objective of the sensitivity analysis is to find the saliency of each feature individually. Except the feature under consideration all features are assigned the mean of their values while training the neural network over the current feature. This process is repeated for all features. Then a random phantom feature is used to compare the saliency of all features to its saliency, thus each feature with saliency less than the phantom's one will be discarded. This method suffers of the massive computations since it trains the neural network to find the saliency for all features, which implies a number of trainings that equal to the number of features.

In his approach, Zhang suggested an evolutionary combination between neural network and genetic algorithm. This method performs the features selection process and the network optimization simultaneously to produce a neural network based classifier. [9] Actually the main disadvantage here is the large amount of computations to end up with the classifier.

3. Problem Definition

Multilayer feed forward neural networks are robust to noise learning machines that perform classification and pattern recognition tasks. Previous experiments, mentioned in section II, showed that the performance of a neural network over a dataset is affected by that data set features. How many features and which features should be presented to the neural network are two critical factors affecting the performance. Since irrelevant features have negative effect on a neural network's classification and generalization ability, removing such features will increase accuracy and efficiency, saves resources, and critically reduce the required training time. Several approaches have been proposed to sieve and select a set of features that match a neural network based classifier, as mentioned in the previous section. Some of those approaches combine the process of features selection and neural network topology optimization together (wrappers) while some of them do not. Even though such combination comes at price by increasing the required amount of computations, it produces more efficient and effective classifiers. Some of the methods that try to overcome the problem of using neural networks over high dimensional datasets require a long training time, while most of them need a large amount of computations. The methods which avoid the long training time perform the entire features selection process or, at least part of it, independently of the neural network optimization, which in turn exacerbates the problem of matching features with learning machine.

3.1 BFSW: Binary Features Selection Wrapper

The proposed method is considered as a wrapper method since it involves both the features selection process and neural network optimization process.

BFSW tries to find a fully connected feed forward neural network over the lowest possible number of features in a dataset with good classification accuracy. As figure 1 illustrates, BFSW starts by training the neural network over the entire candidate features in the dataset to produce a decreasing ordered features ranked list. After that, the same network will be trained over *random probes*, to generate a threshold that separates relevant from irrelevant features.

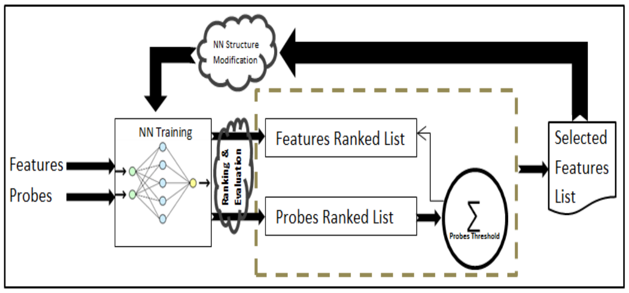


Fig. 1
The General Process of BFSW

3.2 Relevance Index

Although BFSW combines and performs the process of features selection and neural network optimization simultaneously, it avoids the complicated calculus, the long training time, and the massive amount of computations that appear in the methods those have been mentioned earlier. BFSW exploits the weights behavior of the neural network during the training session over the candidate features. Those weights can be understood as indicators of the importance of a specific feature and its contribution to the target output. Note that this is true if all features are normalized into the same range. Thus, after training the neural network over the entire dataset, it is possible to calculate the *relevance index* of each feature. A feature's relevance index S_i is "*relevance quantitative assessment of a candidate feature and the target output*". [10] S_i is calculated according to a feature's final weights which connect it to the output layer through the hidden layer, when that neural network converges, as equation 1 shows [13]:

$$S_i = \sum_{o \in O} \sum_{j \in H} |w_{oj} w_{ji}| \qquad (1)$$

H, O denote the hidden and the output layers respectively. While i is a node (*i.e.* feature) in the input layer. The inner term is the product of the weights from input unit i to hidden unit j, and from hidden unit j to output unit o. The sum of the absolute values of those products over all connections –in the network- from unit i to unit o is the relevance index of feature i that shows its contribution and importance to the output.

In BFSW, to calculate the relevance index of all features in a dataset, we need to train the neural network over the entire set of features only once, which saves training time.

After calculating the relevance index of all features, a decreasing ordered ranked list of the features is produced. Features with high relevance indices of that list (*i.e.* highly informative about the target) are considered as relevant features, while those with low relevance indices (*i.e.* poorly informative about the target) are considered as irrelevant. The problem is that there is not a specific boundary that separates relevant features from irrelevant features in that list. Thus, it is necessary to find a suitable threshold that separates those two parts from each other. That's accomplished using random probes.

3.3 Random Probe Threshold

One of the techniques that can be used to calculate a threshold is the random probes technique. This technique was first suggested in [11]. Random probes are irrelevant features could be generated by shuffling the candidate features vectors in the matrix of training data. [12] This shuffling is done by keeping the targets as they are and randomly swapping candidate features vectors. This results in irrelevant features vectors for the targets. This process keeps the features patterns as they are while producing inconsistency with the target values in comparison with the original dataset. After generating the probes, they will be presented to the same neural network that was used in ranking the candidate features. When the classifier converges over the probes, the relevance indices for all probes will be calculated using equation 1 exactly as what was done then with the candidate features.

The random probe threshold will be generated from the relevance indices of the probes. The random probe threshold is the average of all random probes relevance indices, and it is calculated by using equation 2:

$$P_t = \left(\left(\sum_{i \in I} p_i \right) / \, n \right) \qquad (2)$$

P_t is the probe threshold, i is an input node in the input layer I, while p_i is the relevance index of random probe i which is equivalent to S_i in equation 1 after applying that equation over the final weights of the random probes instead of the candidate features. What equation 2 does, is calculating the average of the probes relevance indices over the total number of the candidate input features which is denoted as n, and thus producing the random probe threshold P_t.

Since a neural network is robust to noise and adaptive in nature, it will converge over the probes even though they are irrelevant features. Hence the random probe threshold will appear somewhere in the candidate features ranked list. This probe is used as a boundary that separates relevant features of the decreasing ordered ranked list from irrelevant features. Thus all features with relevance indices greater than the probe threshold will be kept, while those with lower relevance indices will be discarded. As illustrated in figure 1, after producing the set of the selected features (*i.e.* relevant features) the network structure will be optimized and retrained once again over the selected features. The performance of the new classifier over the selected features will be compared to the performance of the previous one. This process is repeated till the stopping criterion is satisfied. The stopping criterion of this method is based on the classifier's performance over unobserved examples, which is assessed after each training session, directly after classifier structure optimization and the training of the new classifier. Thus, as long as the model performance increases (*i.e.* gets better) in comparison with the performance of the model from the previous training, the process of network training and features reduction goes on. When the classifier performance decreases, the process terminates and the NN classifier with the best performance is chosen.

BFSW takes the structure optimization of a neural network into consideration during the process. At each training session the number of the units in the hidden layer should be twice the number of input units in the input layer, while the number of input units is always equivalent to the number of the selected features at that training session (*i.e.* each input unit receives only one input feature). Figure 2 illustrates the general representation of the hypotheses in the hypothesis space which contains every possible neural network based classifier.

The proposed method is not as straightforward as it seems. Actually an important question arises, which is: *what if the new classifier (after features selection and structure optimization) does not have a better performance than that of the previous one?* The answer to this question has a critical effect on the overall performance of this method. Till now, such a case forms the stopping criterion of the process as mentioned earlier. However we believe that it is not wise to terminate the process at that point, because such an approach does not guarantee that the classifier from the previous training session is the best possible one that could be produced by BFSW, better classifiers could be produced. Thus, we will use a features selection heuristic to circumvent this issue which is: "*individually irrelevant features may become relevant when used in combination*". [4]

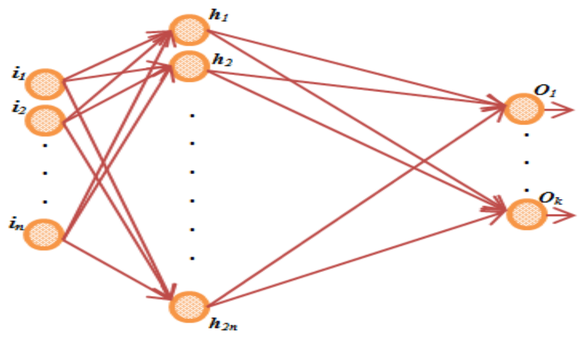


Fig. 2
Neural Network Based Classifier

According to such assumption the proposed method should not terminate just because the new classifier has not better performance than that of the previous one. What we need to do is to control the stopping criterion to assure that the produced classifier is the best possible one that could be produced by BFSW, taking into account that this method does not perform a comprehensive search through the hypothesis space. It is possible to achieve this by making the method more dynamic and directing the search process to reach better solutions.

BFSW tries to find good classifier by applying the "Best First" search technique which can "*back-track to more promising previous subset of features and continue the search from there when the path being explored begins to look less promising*". [15] Every hypothesis in the hypothesis space, as it is illustrated in figure 2, is a fully connected feed forward neural network in which the number of the hidden units is always twice the number of input units which are equivalent to the number of candidate features at the current training session. The hypothesis space is relying on a very useful ordering structure, which is: a Specific-to-General ordering of hypotheses. The most specific hypothesis (*i.e.* the null hypothesis) is that receives the entire set of candidate features as inputs.

The search process starts by enumerating the most specific hypothesis. Then and according to its performance the search direction is directed to those hypotheses which may have better classification accuracy.

Assume that the n-classifier (*i.e.* the produced classifier at training session n) has a better performance than that for $n+1$-classifier. At that point the current stopping criterion would be satisfied, which implies that the process should terminate and the best classifier is the n^{th} classifier. Since the discarded features at session n caused the performance of the produced classifiers to decrease, applying the “Best First” search technique is considered as a reasonable choice because it has the ability to back-track to explore more promising subsets. Since the classifier performance is decreased after discarding a specific set of features, the search process should be applied as described earlier to sieve features in the discarded features set at training session *n* and manipulate it as all features sets have been manipulated in the previous training sessions. Thus, the discarded features set at training session (*n*) will be divided into two parts according to its corresponding probe threshold that is generated by its corresponding discarded probes set (taking into account that the random probe threshold at any training session divides the probes list into two parts as well as it does with the candidate features set). Hence, the upper part of that list will be chosen and appended to the originally selected one to form together the extended selected features set. If the classifier performance over the extended selected features set outperforms that of the previous classifier (*i.e.* classifier produced at training session *n*), then the process proceeds over the selected features set as explained before. Otherwise the extension and the search process should be applied over the discarded features set once again. This process goes on while the discarded features set contains more than one feature and the performance of the produced classifier is still decreasing, at that situation the process terminates and the classifier with the highest performance is eventually chosen among all produced classifiers.

Even though the search strategy here does not enumerate every possible hypothesis in the hypothesis space either directly or indirectly, it is efficient since it avoids the exhaustive search which consumes long time, and eventually produces an efficient classifier.

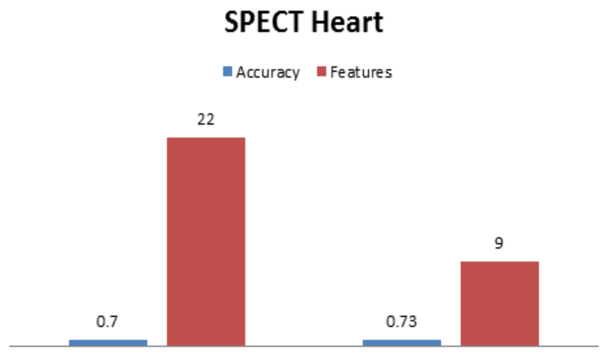
4. Implementation and Results

BFSW was implemented and some preliminary experiments ran in order to assess the effectiveness of this combination. The classifier is a multilayer feed forward neural network in which the number of the units in the input layer is equivalent to the number of the candidate features at the current training session, while the units’ number in the hidden layer is always twice the number of units in the input layer. The classifier is trained over the candidate features, the random probes, and the selected features using backpropagation learning algorithm and the same weights initial values those were used at the first training session. Actually, the neural network is trained over the random probes only once (at the first training session) and the produced relevance indices are used during the rest of the training sessions without any need to retrain the network over them once again.

The training and testing processes were performed over the SPECT Heart and Breast Cancer datasets, real problems. [14] Figure 3 shows the results of the BFSW implementation over those datasets.

Figure 3 illustrates that the entire candidate features are 31 for the Breast Cancer and 22 for the SPECT Heart. In both datasets BFSW was able to recognize irrelevant and redundant features and make a decision to rule them out while keeping the most informative (*i.e.* relevant) features by using the random probe threshold and the classifier weights as relevance indices. It is possible to note that the number of features was roughly reduced from 22 to 9 and 31 to 7, for SPECT Heart and Breast Cancer respectively while the classifier overall classification accuracy was improved. The classification accuracy increased after features selection process.

(a)



(b)

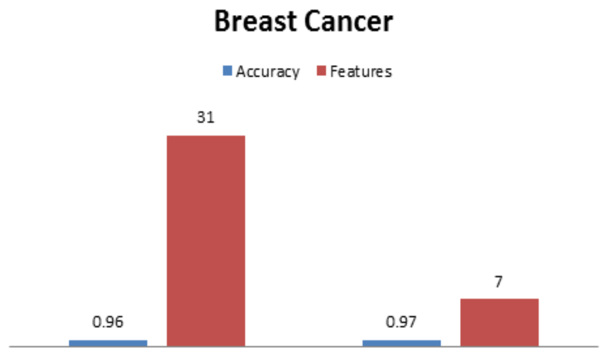


Fig 3:
Implementation Results over SPECT Heart and Breast Cancer Datasets

For SPECT Heart the classification accuracy improved to 0.73 over 9 features instead of 0.7 over 22 features, and it was also improved for the Breast Cancer to be 0.97 over 7 features instead of 0.96 over a set of 31 features. Even though the classification accuracy was slightly increased, it is better to have 0.97 or 0.73 of classification accuracy over a small set of features than having lower or even the same classification accuracy over the entire set of features. During the implementation we noticed that BFSW consumed only one training session to converge to global maxima over the

SPECT Heart, while for Breast Cancer it consumed two training sessions. This variation appears because each data set has its own characteristics, patterns, and correlations which affect the performance of BFSW and make it vary over different datasets. Moreover BFSW is considered as a random technique, since it is based on neural networks which are initialized randomly and the random probes which are generated randomly. Such factors interpret the variation of the BFSW over different datasets.

An important advantage of this method is the simplification of random probes usage. As said before, random probes approach was first suggested in [11]. The major aim of this approach is to reduce the number of the candidate features independently of the learning machine, thus it is considered as a filter approach. In order to make sure that the selected features, after applying random probes, are sufficient to perform the learning task, the decision of discarding features is supported by a statistical test. More details about traditional usage of random probes are available in [4]. BFSW used the random probes in a different and simple way since it makes the decision of evaluating and discarding features directly related to the learning machine performance which in turn helps to avoid the required statistical tests. Random probes are normally used with the Gram-Schmidt technique which was first suggested in [7]. BFSW replaced Gram-Schmidt with the neural network and used its weights as relevance indices for the candidate features. This way makes the relevance indices more informative about the effect and the importance of each candidate feature.

5. Conclusion and Future Work

To sum up, the proposed method is a wrapper method that aims to find sufficient features subset that is convenient to match a neural network based classifier by searching a hypothesis space which has a naturally occurring Specific-to-General ordering structure. The search process is implemented under the following assumption: "*the solution exists in the hypothesis space*". The major aim of BFSW is to simplify and speed up the search process to reach the required hypothesis in the hypothesis space, by applying an efficient search technique than those used in some of the currently existing methods. This method is a novel one since it uses random probes in a wrapper technique and combines it directly with the learning machine. In BFSW the neural network topology is taken into consideration and it is optimized at each training session. The preliminary results of BFSW are promised results and showed the possibility and the efficiency of combining random probes with neural networks. In the future work, we are to implement BFSW with more challenging datasets and compare its results with the currently existing methods.

6. References

[1] T. Mitchell, *Machine learning*. Singapore: McGRAW-HILL, 1997.

[2] M. Negnevitsky, *Artificial intelligence: a guide to intelligent systems*. Britain: Addison-Wesley, 2005.

[3] H. Yuan, S. Tseng, W. Gangshan, and Z. Fuyan, "A two-phase feature selection method using both filter and wrapper," *IEEE International Conference*, vol. 2, 1999, pp. 132 – 136.

[4] I. Guyon, M. Nikravesh, S. Gunn, and L. A. Zadeh, *Feature extraction – foundations and applications*. Berlin Heidelberg New York: Springer 2006.

[5] M. Yacoub and Y. Bennani, "HVS: a heuristic for variable selection in multilayer artificial neural network classifier," *Artificial Neural Networks in Engineering*, 1997, pp. 527-532.

[6] M. Stricker, F. Vichot, G. Dreyfus, and F. Wolinski, "Two-step feature selection and neural network classification for the trec-8 routing," in *Proc. of the Eight Text Retrieval Conf*. 1999. http://arxiv.org/ftp/cs/papers/0007/0007016.pdf

[7] S. Chen, A. Billings, and W. Luo, "Orthogonal least squares methods and their application to non-linear system identification," *International journal of control*, vol. 5, 1986, pp. 1873-1896.

[8] M. J. Embrechts, F. Arciniegas, M. Ozdemir, C. M. Breneman, K. Bennett, and L. Lockwood, "Bagging neural network sensitivity analysis for feature reduction for in-silico drug design," in *IEEE International Joint Conf.*, 2001, pp. 2478–2482.

[9] B. Zhang, "A Joint evolutionary method based on neural network for feature selection," in *IEEE Second International Conf. on Intelligent Computation Technology and Automation*, 2009, pp. 7 – 10.

[10] H. Stoppiglia, G. Dreyfus, R. Dubois, and Y. Oussar, "Ranking a random feature for variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, 2003, pp. 1399-1414. http://remidubois.free.fr/publications/118.pdf

[11]H. Stoppiglia, "Methodes statistiques de selection de modeles neuronaux, application financieres et bancaires," PhD. Dissertation, Universite Pierre et Marie Curie, Paris, 1997.

[12] J. Bi, K. P. Bennett, M. Embrechts, C. M. Breneman, and M. Song, "Dimensionality reduction via sparse support vector machines," *Journal of Machine Learning Research*, vol. 3, 2003, pp. 1229-1243. http://homepages.rpi.edu/~bennek/papers/bi03a.pdf

[13] P. Leray, and P. Gallinari, "Feature selection with neural networks," *Behaviormetrika*, vol. 26, 1999, pp.145–166.

[14] "Machine Learning Repository", http://archive.ics.uci.edu/ml/

[15] M. A. Hall. "Correlation-based Feature Selection for Machine Learning." PhD, The University of Waikato, Hamilton, NewZealand, 1999.