Feature Subset Selection Based on ANN Sensitivity Analysis – a Practical Study

J. N. FIDALGO
INESC Porto - Instituto de Engenharia de Sistemas e Computadores
Pr. da Republica, 93, 4050 – 497, Porto
PORTUGAL

Abstract: - Feature subset selection is a central issue in a vast diversity of problems including classification, function approximation, machine learning and adaptive control. On a wide variety of applications, especially when using real data, input features may be not independent and output variable depends on the relationship among inputs rather than on input values themselves. Feature selection methods that assume independence of attributes will fail on these cases. On the other side, most of alternative approaches are quasi-exhaustive, requiring large CPU processing time. In this paper, an alternative methodology based on sensitivity analysis of trained artificial neural networks (ANN) is analyzed. Results so far attained on illustrative toy examples and on real data support the validity of the developed approach.

Key Words: - feature selection, neural networks, sensitivity, correlation

1 Introduction

Feature subset selection (FSS) consists of identifying a subset of significant attributes, discarding the remaining ones, to represent adequately the system state, initially characterized by a larger set of redundant features. FSS is a essential task when using a wide variety of tools like artificial neural networks (ANN), fuzzy sets, the k-nearest neighbors method or regression trees for classification, function approximation, machine learning and adaptive control

When the number of attributes is small, exhaustive or quasi-exhaustive search may be used to select the best attributes set in order to accomplish the desired task. But the number of possible combinations grows quickly with the number of attributes – the *curse of dimensionality*. As higher is the number of attributes the faster and straitforwarder should be the FSS approach. Typically, FSS methods that assume independence of attributes like correlation analysis, F measure, or information gain, are simple and fast, but may fail on a wide sort of applications. On the other side, there are the FSS time-consuming approaches, using quasi-exhaustive or genetic like search, or needing the repeated training of ANN, while less significant features are discarded one by one [4-7]. Besides, some FSS techniques can only deal with binary inputs and/or outputs.

This paper describes an alternative FSS approach that is simple, straitforward, and, as far as our examples had shown, has no hard application limitations.

2 A Failing Example: Rank Correlation

A common FSS approach adopted frequently is based on computing a input marginal importance measure, i.e., considering each input in isolation. As an example, if rank correlation method is adopted, the following algorithm is followed:

a) Computation of the individual correlation's factors (x_i; y),where x_i represents possible feature i (i=1,..,number of features) and y is the output;
b) Ranking input variables according to its correlation factor with respect to the output;
c) Selection of the N variables with highest values.

As a matter of illustration of how this algorithm mail fail, consider Table 1, where the variables x_i where randomly generated in the interval [-2; 2]. 2000 random patterns were produced. The output variable y was computed for each input pattern (x1,..,x8) using the following equation:

$$y = \frac{1}{(x_2 - x_4)^2 + 1} + x_6 \sin(2x_8) \qquad (1)$$

Table 1 – Sample of generated training set

x1	x2	x3	x4	x5	x6	x7	x8	y
0.31	1.68	-1.46	1.37	-0.56	-0.72	1.49	-0.03	1.72

Equation (1) shows that output variable y does not depend on x_1, x_3, x_5 and x_7. Table 2 results from the calculus of a correlation analysis and ranking the variables according to the absolute value of the correlation indexes.

Table 2 – Ranked correlation modules indexes

Ranking	Variable	Correlation index
1	x4	0.043
2	x1	0.042
3	x5	0.025
4	x6	0.014
5	x8	0.012
6	x3	0.008
7	x7	0.006
8	x2	0.002

Variables x1 and x5, that were not used in the computation of y, appear in the group of 4 highest correlated with output. These results prove that, at least for a certain type of functions, correlation analysis does not provide correct answers on FSS phase. It is clear, when analyzing function (1), that there are variables intimately coupled (*e.g.*, x_2 and x_4). FSS methods based on measures that relate individually each input x_i with output y may fail.

3 Feature Subset Selection Based on ANN Sensitivity Analysis

Now, let's experience another approach. Our proposal comprises the following steps:

a) Train an ANN to learn y function, using all possible candidate features. ANN will present an output value O that should be close to y. As before, we suppose we don't know which variables are important or not;
b) For all training patterns, compute $\partial O/\partial x_i$, that is, the derivative of the output with respect to each input i. Later on this paper, we describe a simple algorithm for computing these vales;
c) Compute the mean absolute value of derivatives for each input, defining our sensitivity index s_i:

$$s_i = \frac{\sum_{p=1}^{\text{nr.of patterns}} \left| \frac{\partial O}{\partial x_i} \right|_p}{\text{nr.of patterns}} \tag{2}$$

where p represents the pattern index.

Why should one expect this index give something useful? Suppose we compute the output freezing all input variables except x_a and we put the result in a graphic. The same operation is repeated to variable x_b. If tangent along the curve $f(x_b)$ presents generally a higher slope than for $f(x_a)$, then:

$$\left| \frac{\partial f}{\partial x_b} \right|_{\text{mean}} > \left| \frac{\partial f}{\partial x_a} \right|_{\text{mean}} \tag{3}$$

Following the definition of sensitivity given by equation (2), one can state that $s_b > s_a$. Concluding, if a robust technique might be used for computing these indexes, we'll have an alternative methodology for FSS.

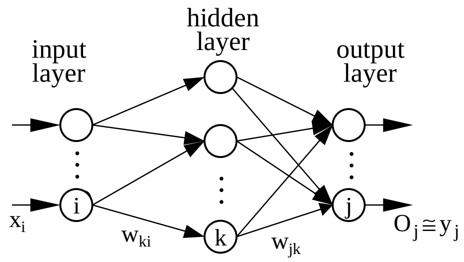

Fig. 1 - ANN general feedforward architecture

Computing these derivatives using a trained ANN is in fact quite simple. Suppose Fig. 1 represents the ANN trained to approximate a given function **y**=f(**x**), where **y** and **x** represent, respectively, output and input vectors. During learning, ANN weights are changed such way its outputs O_j get closer and closer to the targets y_j. Each unit of hidden or output layers comprises two functions: a weighted sum and a transfer function:

$$n_k = \sum_i w_{ki} x_i \tag{4}$$

$$O_k = f_k(n_k) \tag{5}$$

Transfer function are generally sigmoid type function, like hyperbolic tangent, for hidden layer(s) and linear for output layer. Input units just distribute input values x_i.

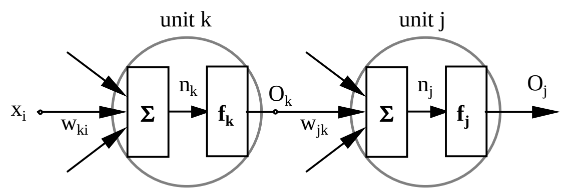

Fig. 2 - ANN "one unit per layer" scheme

Fig. 2 shows just one unit for each layer of the ANN. The derivative of the output O_j with respect to O_k, the output of unit k, is given by:

$$\frac{\partial O_j}{\partial O_k} = f'_j \, w_{jk} \tag{6}$$

where f'_j is the derivative of f_j at point n_j. Using a chain rule:

$$\frac{\partial O_j}{\partial x_i} = \sum_k f'_k \, w_{ki} \frac{\partial O_k}{\partial x_k} \tag{7}$$

where index k represents all units fed by input x_i. If the ANN is a fully connected one, index k refers to all units in the hidden layer. This simple and elegant procedure is similar to the one used for weights adaptation on Backpropagation Algorithm [1,2]. Table 3 shows the s_i indexes obtained by applying formula (2). Dummy variables (x1, x3, x5 and x7) present the lowest s_i values showing that proposed methodology was successfully on this example.

Table 3 – ANN sensitivity indexes

Ranking	Variable	s_i index
1	x8	1.625
2	x4	0.893
3	x2	0.879
4	x6	0.685
5	x5	0.087
6	x7	0.035
7	x1	0.019
8	x3	0.015

3.1 Some more details about used ANN

The 2000 patterns were normalized to have zero mean and a standard deviation of one. This removes of offset issues and measurement scales. 1500 patterns were used for training and 500 for testing. Training algorithm was the Adaptive Backpropagation [3]. A description of ANN architectures experienced is reported on a further section on this paper.

4 Sensitivity Analysis Under Noise

Results presented before concern to a clean well defined mathematical function y=f(x1, .., xN). That is not the case of real world applications. In fact, on practical applications, there's no "function" and data is affected by noise caused by errors on measuring, failing of data acquisition system, recording problems, and so on. Being so, it is licit to suspect that presented sensitivity calculus approach may fail under these circumstances, that is, when one has just a collection of points to be dealt with to perform a multiregression task.

In the following analysis, random noise has been added to function variables under the succeeding hypothesis:

a) 10% of noise added to output y;
b) 20% of noise added to output y;
c) 20% of noise added to both inputs (x1,.., x8) and output y.

As ANN output has been normalized to have zero mean and a standard deviation of 1.0, one may roughly admit that output values are mostly contained in the interval [-1.0;1.0], and consider 10% of noise when random noise is generated in the interval [-0.1;0.1]. That's what was considered on case a). On case b), random noise in the interval [-0.2;0.2] was added to output. Finally on case c), random noise in the interval [-0.2;0.2] was added to all inputs and output. Results obtained are presented on Table 4, were performance refers to the mean absolute percentage error. ANN was trained during 5000 epochs in all cases. Sensitivity indexes are calculated using all patterns of the training set. As can be observed, the ranking of s_i allows, in all cases reported, to select the real features (x_2, x_4, x_6 and x_8), with the highest index values. Despite the degradation of ANN performance as noise increases (note more than 20% error on c) case), s_i ranking still provides a correct emplacement of features, with the dummy variables having the lowest values. At the same time, as noise level increases, the differences on s_i values among real features and dummy ones become smaller, which is also an expected result.

One may also expect that if noise too high, one might no more obtain a correct ranking of s_i. But, in this case, the ANN performance will also be poor. As conclusion, it seems that if ANN is able to perform reasonably, it can also provide the correct s_i ranking, which is a very interesting result.

5 S_i Calculus With Smaller Datasets

The number of patterns (1500) used for training on previous studies with 8 potential input features may be a little too much when compared to some real applications, when the number of training examples may be scarce. So, the analysis were repeated but using this time the second set (the previous test set with 500 patterns). Results obtained are presented on Table 5.

Note that performance error on test set increases from epoch 4000 to 5000 showing that overfitting is

Table 4 – Sensitivity indexes calculus under noise

	Performance (%)		ANN sensitivity indexes							
case	training (1500)	test (500)	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
a)	4.97	5.73	0.004	0.875	0.005	0.877	0.005	0.665	0.004	1.590
b)	9.22	10.31	0.009	0.881	0.013	0.883	0.008	0.675	0.005	1.592
c)	23.45	26.90	0.259	0.880	0.032	0.831	0.321	0.691	0.060	1.557

Table 5 – Sensitivity calculus on a smaller training set

	Performance (%)		ANN sensitivity indexes							
epoch	training (500)	test (1500)	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
3000	3.84	5.59								
4000	3.54	5.09								
5000	3.46	5.16	0.009	0.796	0.008	0.794	0.008	0.681	0.010	1.554

occurring. Even so, s_i ranking still provides the wanted results. If dummy features are eliminated and training is repeated using only the remaining ones, the performance errors attained after 2500 training epochs are 3.11% on training set (500 patterns) and 3.66% on test set (1500 patterns). These results, showing smaller errors on both training and test sets and obtained after fewer training epochs, confirm the advantages of eliminating dummy variables that just introduce noise.

6 Dependence on ANN architecture

Previous results refer to an ANN architecture (set initially by chance) of 8-10-6-1: *i.e.*, 8 inputs, 10 units in the first hidden layer, 6 units in the second hidden layer and 1 output unit. The study described on this section aims to infer the possible limitations of ANN architecture choice on the calculus of sensitivity indexes. First, we analyze how ANN architecture simplification may affect the results. After, we go on the opposite direction increasing the number of ANN parameters and interpreting the results. Table 6 presents the performance errors and sensitivity indexes for several ANN architectures. With the exception of ANN 8-2-1, that provide a bad ranking of s_i, all the other perform satisfactory under this issue. The performance error for the 8-2-1 case is about 60%, which is really a poor approximation to the desired function. Next case, 8-4-1, despite the still poor 50% error, already provides a correct ranking of s_i. The next 2 architectures , 8-20-10-1 and 8-50-20-1, provide good results in both performance and s_i ranking issues. The last case reported, on bottom of Table 6, also concerns to ANN 8-50-20-1, but this time a smaller training set (with 500 patterns) was used. A larger number of epochs was also considered in order to force the growth of the overfitting phenomena – note the 0.3% of error in the training set against 33.2% in the test set. In spite of overfitting, s_i ranking is still suitable. Note also that derivatives are generally higher because excessive training leads to higher non-linearity mapping surfaces.

7 Some More Testing

7.1 Equality relations among inputs

Similar procedure was performed with the following function:

$$y = \frac{1}{(x_2 + x_4 - x_6 - x_8)^2 + 1} + \\ + \sin(x_{10} + x_{12} + x_{14} + x_{16}) \quad (8)$$

All input variables, except x_1, x_{17}, x_{18}, x_{19} and x_{20} were randomly generated within the interval [-2; 2]. The following relations were established:

$$x_1 = 3x_2 \quad x_{17} = x_4 \quad x_{18} = x_{19} = x_{20} = x_6 \quad (9)$$

First, a correlation analysis was performed to eliminate correlated variables. This step is necessary when using the proposed FSS approach because sensitivity indexes may be distorted. Note, for instance, that function (8) depends on x_6 and not on x_{18}, x_{19}, and x_{20}, and it is impossible just by analyzing data patterns to infer that conclusion. As $x_{18}=x_{19}=x_{20}=x_6$, the ANN may use indifferently any one of these inputs, depend a lot, for instance, on x_{19} and just a bit on x_6, x_{18}, and x_{20}. So, index s_6, we wish to be included in the set of the highest ones, may be distributed by s_6, s_{18}, s_{19} and s_{20}. This fact is probably the main argument against FSS based on sensitivity analysis [10]. However, this problem can be easily overwhelmed by using a simple technique like correlation analysis for discarding linked variables before proceeding with the s_i calculus. In this case, correlated features (x_2, $x_{17\text{-}20}$) were discarded. The application of the proposed FSS approach to the remaining variables provides the results summarized on Table 7. s_i indexes were normalized to give the sum one. This way relative s_i values that can be perceived as percentages of the whole. A gray shadow marks dummy variables. In this case, after 4000 training epochs, real attributes have already the highest s_i values, despite of the difference between s_1 (the real feature with lowest s_i) and s_{15} (the dummy feature with highest s_i) is not relevant. However, as long as training proceeds, one can witness the growth of that difference.

Table 6– Sensitivity calculus on different ANN architectures

ANN architecture	Performance (500 epochs) (%) training (1500)	test (500)	ANN sensitivity indexes s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
8-2-1	59.0	62.3	0.023	0.747	0.031	0.732	0.031	0.034	0.024	0.005
8-4-1	49.5	51.4	0.020	0.800	0.031	0.802	0.026	0.398	0.039	0.735
8-6-1	21.3	22.3	0.009	0.803	0.015	0.801	0.019	0.591	0.013	1.318
8-20-10-1	6.1	6.5	0.015	0.846	0.012	0.847	0.009	0.660	0.010	1.547
8-50-20-1	3.4	4.7	0.017	0.856	0.019	0.864	0.017	0.664	0.015	1.586

ANN architecture	Performance (1100 epochs) (%) training (500)	test (1500)	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
8-50-20-1	0.3	33.2	0.255	0.901	0.222	0.911	0.241	0.724	0.276	1.558

Table 7 – s_i variation with training epochs

4000 epochs		5000 epochs		6000 epochs	
i	**s_i**	**i**	**s_i**	**i**	**s_i**
10	0.163	10	0.170	10	0.171
16	0.120	16	0.147	16	0.156
12	0.113	12	0.144	12	0.155
14	0.110	14	0.143	14	0.154
4	0.069	4	0.070	4	0.071
8	0.064	8	0.068	8	0.070
6	0.064	6	0.068	6	0.070
1	0.059	1	0.066	1	0.068
15	0.057	15	0.029	15	0.020
5	0.042	5	0.021	5	0.015
9	0.042	9	0.022	9	0.015
13	0.028	13	0.015	13	0.010
7	0.026	7	0.014	7	0.010
3	0.022	3	0.011	11	0.008
11	0.021	11	0.010	3	0.008

7.2 Functional relations among inputs

In this study, function to be analyzed is still given by equation (8), but relations among variables are given by the following equations:

$$\begin{cases} x_1 = 3x_2 \\ x_{17} = x_4\, x_2 - 2 \\ x_{18} = x_4 \sin(x_{10} - x_{12}) \\ x_{19} = x_{20} = x_6 \end{cases} \qquad (10)$$

This FSS problem is harder than previous one because correlation analysis may not eliminate all functional related variables. In this case, only eliminates variables x_2, x_{19} and x_{20}. As $x_1=3x_2$, it is equivalent to eliminate either x_1 or x_2, despite of x_2 appear on equation (8) and x_1 not. The remaining 17 variables were used as inputs of a new ANN. Table 8 shows s_i results obtained after 1500 training epochs. The performance attained was 2.6% on training and 2.8% on test sets.

Table 8 – s_i indexes ranking for eq. (8) and (9)

Ranking	Variable	s_i index
1	x14	0.1773
2	x12	0.1761
3	x10	0.1750
4	x16	0.1146
5	x8	0.0975
6	x4	0.0850
7	x6	0.0841
8	x1	0.0836
9	x18	0.0026
10	x17	0.0011
11	x13	0.0008
12	x15	0.0006
13	x5	0.0005
14	x11	0.0004
15	x7	0.0004
16	x3	0.0004
17	x9	0.0004

7.3 Friedman series

This example is based on Friedman series [12]:

$$y = 10\sin(x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + N(0,1) \qquad (11)$$

where N(0,1) represents normal distributed noise. Data set has ten input variables $x_1,..,x_{10}$ but the response only depends on $x_1,..,x_5$. Performance obtained in this case was 3.1% on the training and 4.4% on test sets. Table 9 shows s_i indexes obtained. In [13] authors conclude that FSS methods based on sensitivity analysis fail on this series. Table 9 shows that isn't true: dummy variables have the lowest s_i.

Table 9 – s_i indexes ranking for eq. (8) and (10)

Ranking	Variable	s_i index
1	x4	0.2913
2	x3	0.2662
3	x5	0.1456
4	x2	0.1317
5	x1	0.1207
6	x8	0.0151
7	x6	0.0132
8	x10	0.0060
9	x7	0.0058
10	x9	0.0046

8 S_i Analysis On Logic Functions

Suppose we train an ANN to emulate function (12):

$$\begin{aligned} &\text{if } ((x_2 < -0.5) \vee (x_4 > 0)) \wedge ((x_6 < -0.5) \vee (-1 < x_8 < 1)) \\ &\quad y=1 \\ &\text{else } y=0 \end{aligned} \qquad (12)$$

where $\vee$ stands for logic OR and $\wedge$ for AND.

If ANN units transfer functions are continuous, the result is a continuous function (the ANN), trying to imitate a no-continuous one. It would still be possible to compute the <u>finite</u> derivatives of the output with respect to inputs. The same approach used for the first example was applied here. Results are shown on Table 10:

Table 10 – s_i indexes for function (12) and (13)

	Function (12)		Function (13)	
Ranking	Variable	s_i index	Variable	s_i index
1	x6	1.192	x4	0.2846
2	x2	1.166	x2	0.2782
3	x8	0.962	x6	0.2236
4	x4	0.505	x8	0.2108
5	x5	0.086	x5	0.0009
6	x1	0.066	x1	0.0008
7	x7	0.060	x7	0.0006
8	x3	0.050	x3	0.0005

Another study was carry out using the same patterns generated for function (12) but resolving first the inner parenthesis. For instance,

$$\begin{aligned} &\text{if } (x_2 < -0.5) \text{ then } x_2 = 1 \\ &\text{else } x_2 = 0 \end{aligned} \qquad (13)$$

Dummy variables were set to 1 if greater than zero; and zero otherwise. The s_i attained are also shown on Table 10. Some authors, like in [11], argue that derivatives are not suitable for discrete inputs. This is surely true for non-continuous ANN, like when units have a step for transfer function. However, as a continuous ANN is being used, it is mathematically sensible to compute derivatives, even when one is trying to reproduce a non-continuous function.

9 FSS on A Real Data Case

This analysis is based on data of power system from Crete, Greece. Patterns data was generated by changing several system parameters and computing simulation of a security measure [8,9], when power system is shaken by a given disturbance d_1. The whole process was repeated for a second disturbance d_2. Initial set comprises 60 attributes. A correlation analysis was performed and for each attribute pairs with a correlation index higher larger than 0.90, a variable was discarded. This procedure eliminates 41 variables. An ANN was trained with the remaining 19 and s_i calculus was performed. s_i indexes were normalized to give the sum one and features with s_i index below 0.02 were discarded. This heuristic has eliminated 12 more variables. A new ANN was trained with the remaining 7 features. Performance was computed and compared to older values reported on [8,9], were 22 attributes were used (Table 11).

Table 11 – Results for the Crete case d_1

case d1				case d2			
old perf.		new perf.		old perf.		new perf.	
train	test	train	test	train	test	train	test
0.044	0.043	0.033	0.036	0.059	0.064	0.023	0.034

10 Conclusion

A practical study of an alternative FSS approach based on correlation analysis and ANN sensitivity analysis was carry out. The method is simple, straitforward and fairly insensitive to noise and ANN chosen architectures. It requires some time for the initial ANN training but, unlike some concurrent methods, allows the discarding of several (or even all) dummy features at a time. Results attained so far on function approximation studies support the validity of proposed approach and encourage further developments and tests.

However, its application to time series should be evaluated because each value is strongly correlated with previous one. As the first step of the method consists of deleting all correlated variables, it means that all variables would be deleted except one. Then, each forecast would be only dependent on previous one, what is, for sure, not consistent with experience. Notice however that on real time series like load forecasting, noise display an important role. In this case, it is important to use several correlated inputs in order to decreased sensitivity to this factor. Nevertheless, further developments on the method should be made to overcome this limitation.

References:

[1] D. E. Rumelheart et. al., "Learning Internal Representations By Error Propagation", In Parallel Distributed Processing, Mit Press, Cambridge, Ma, USA, 1986

[2] Simon Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed., Prentice-Hall, 1999

[3] F. M. Silva, L. B. Almeida, "Acceleration Techniques For The Backpropagation Algorithm", In *Neural Networks*, L. B. Almeida and C. J. Wellekens (Eds.), Springer-Verlag, 1990

[4] L.M.Belue and K.W.Bauer, “Determining Input Features For Multilayer Perceptron”, *Neurocomputing*, Vol. 7, no. 2, 1995

[5] Rudy Setiono, Huan Liu, “Neural Network Feature Selector”, *IEEE Transactions on Neural Networks*, Vol. 8, no. 5, 1997

[6] Jihoon Yang, V. Honavar, “Feature Selection Using Genetic Algorithm”, *Proc. of Genetic Programming Conf.*, GP’97, Stanford Univ., 1997

[7] Haleh Vafaie, Keneth De Jong, “Robust Feature Selection Algorithms”, In Proceedings of the 5th *IEEE In. Conference on Tools for Artificial Intelligence*, Boston, MA: IEEE Press, 1993

[8] Maria Helena Vasconcelos, J. A. Peças Lopes, J.N. Fidalgo, Matt Mitchell, “Report of the INESC-Porto Activities in the Field of Security Assessment within the CARE Project”, *CARE Project Contract JOR3-CT96-0119*, January 1999

[9] J. A. Peças Lopes et.al., “On-Line Dynamic Security Assessment Of Isolated Networks Integrating Large Wind Power Production”, *Wind Engineering Review*, Vol. 23, No. 2, 107-117, 1999

[10] Warren S. Sarle, “How to Measure Importance of Units”, SAS Intitute Inc., Cary, NC, USA ftp://ftp.sas.com/pub/neural/importance.html

[11] W. G. Baxt, H. White, “Bootstrapping Confidence Intervals for Clinical Input Variables Effects in a Network Trained to Indentify the Presence os Acute Myocardial Infarction”, *Neural Computation*, No. 7, 624-638, 1995

[12] Jerome H. Friedman, “Multivariate Adaptive Regression Splines”, *The Annals of Statistic*s, 19(1), 1-141, 1991

[13] Pierre van de Laar, et.al., “Partial Retraining: a New Approach to Input Relevance Determination”, *Int. Journal of Neural Systems*, 9, 75-85, 1999