2006 International Joint Conference on Neural Networks
Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada
July 16-21, 2006

Optimization of Neural Networks with Multi-Objective LASSO Algorithm

Marcelo Azevedo Costa and Antônio Pádua Braga

***Abstract*— This paper presents a bi-objective algorithm that optimizes the error and the sum of the absolute weights of a Multi-Layer Perceptron neural network. The algorithm is based on the linear Least Absolute Shrinkage and Selection Operator (LASSO) and provides simultaneous generalization and weight selection optimization. The algorithm searches for a set of optimal solutions called Pareto set from which a single weight vector with best performance and reduced number of weights is selected based on a validation criterion. The method is applied to classification and regression real problems and compared with the norm based multi-objective algorithm. Results show that the neural networks obtained have improved generalization performance and reduced topology.**

I. Introduction

Neural networks are data driven models capable to fit any function as long as they have enough number of neurons and network layers. One of the most used neural models is the Multi-Layer Perceptron (MLP) [1]. A MLP with one hidden layer with sigmoidal activation function and linear output is capable to approximate any continuous function [2]. However, in practical situations, the data is corrupted with noise and the neural network must learn the true relationship among the input and output patterns minimizing the noise effect.

In gradient descent learning the algorithm approximates locally the error surface by calculating the first derivative of the error. However, the algorithm does not guarantee convergence to the global minimum of error. Moreover, even if the network reaches the global minimum error, it does not imply that the model response is optimal. An optimal solution represents a model whose response is very close to the function that generated the data. This model has the ability to provide reliable outputs for patterns which were not presented to the networks in the training process. For neural networks, the number of weights and their magnitudes are related to its fittness to the data [4].

Basically, a network with large number of weights but with small amplitudes behaves as an underfitted model that gradually overfits data during training. Based on this principle, the early-stopping algorithm [5] splits the data set into training and validation set. The training set is used to calculate the weights update, the validation set is used as a goodness of fit measure in the meanwhile. The algorithm stops when the validation error starts to increase. The weight-decay algorithm [6] adds a penalty term to the error cost function that confines the norm of the weights to certain proportions. This procedure smoothes the model output but it is also sensitive to initial conditions. Pruning algorithms [7] aim at extracting weights after training process as a way to reduce the network complexity. In general, those algorithms try to find an unknown condition for the weight's amplitude that improves the network generalization capability [4].

The Multi-Objective Algorithm (MOBJ) [8] controls the weights amplitude by optimizing two objective functions: the error function and norm function. The algorithm is able to reach a high generalization solution and avoid over and undefitting. A comparison with standard algorithms can be found in [8]. The LASSO approach proposed in this paper aggregates the MOBJ high generalization capability and an automatic weight selection through the specification of an alternative objective function: the sum of the absolute weights. Therefore, it generates networks with reduced number of weights when compared with MOBJ solutions but with similar prediction performance.

This paper is organized as follows. In Section 2 the original multi-objective (MOBJ) algorithm is described. In Section 3 the LASSO proposal is presented and illustrated. In Section 4 some implementation and data sets details are discussed. In Section 5 main results are shown and discussed. Summary and concluding remarks are given in Section 6.

II. The Multi-Objective Algorithm

The original Multi-Objective algorithm (MOBJ) [8] optimizes the error and norm cost functions. Both functions have different minima which means that a solution with both minimum error and norm functions is non-feasible. The Pareto set represents the set of solutions between two extreme solutions: the solution with minimum error (and large norm) and the solution with minimum norm (and large error). Figure 1 shows the Pareto set shape. It represents the boundary between feasible solutions and non-feasible region. Moreover, the Pareto set represents solutions with minimum error subject to specific norm constraints. Based on this concept the overall multi-objective optimization problem can then be written as a constrained mono-objective function:

$$\mathbf{w}^* = arg\ \min \tfrac{1}{N}\sum_{j=1}^{N}\left(d_j - y(\mathbf{w},\mathbf{x}_j)\right)^2 \qquad (1)$$
$$subject\,to:\ ||\mathbf{w}|| \le \eta_i$$

Antônio Pádua Braga, Depto. Engenharia Eletronica Campus da UFMG (Pampulha), Caixa Postal 209, CEP 30.161-970, Belo Horizonte, MG, Brazil (phone: +55 31 3499 4869; fax: +55 31 3499 4850; email: apbraga@cpdee.ufmg.br).

Marcelo Azevedo Costa, Depto. Estatistica Campus da UFMG (Pampulha), Caixa Postal 702, CEP 31.270-901, Belo Horizonte, MG, Brazil (phone: +55 31 3499 5937; fax: +55 31 3499 5924; email: azevedo@est.ufmg.br).

where $\{\mathbf{x}_j, d_j\}_{j=1}^{N}$ are the input-output data set, N is the sample size, $||\mathbf{w}|| = \sqrt{\sum_t w_t^2}$ and η_j is the norm constraint value. For each constraint value η_j, there is one solution associated within the Pareto set.

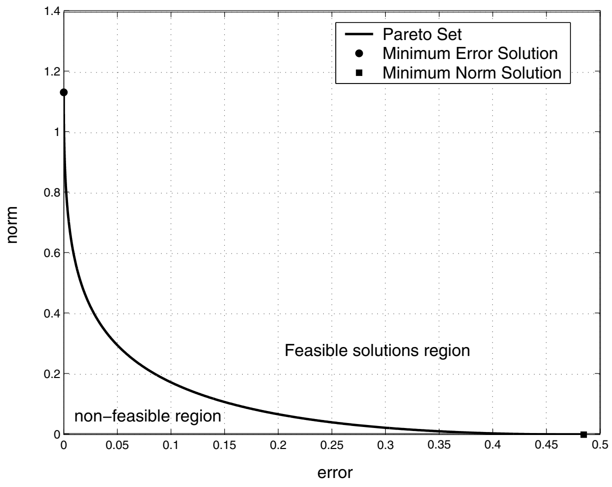


Fig. 1. Pareto Set and the two objective functions minima.

The multi-objective problem can be solved by a constrained optimization method as the *ellipsoidal algorithm* [9]. Alternative multi-objective algorithms include Sliding Mode Multi-Objective algorithm [10], Levenberg-Marquardt based Multi-Objective [11] and Multi-Objective Evolutionary Algorithms [12], [13].

III. The LASSO Method

The *least absolute shrinkage and selection operator* (LASSO) [15] minimizes the residual sum of squares (error) subject to the sum of the absolute weights being less than a constant t.

$$\mathbf{w}^* = arg\ \min \tfrac{1}{N} \textstyle\sum_{j=1}^{N} \left(d_j - y(\mathbf{w}, \mathbf{x}_j)\right)^2$$
$$subject\ to: \ \textstyle\sum_i |w_i| \le t \qquad (2)$$

The method is very similar to the norm constrain approach with subtle but important differences. To illustrate the differences between the MOBJ and LASSO methods, we present a single perceptron with hyperbolic tangent activation function, one input and two weights: the input weight w and the bias weight, b. The perceptron's output equation is: $y(x_i) = tanh(w.x_i + b)$. The following patterns: $(x_i, y_i) = \{(-3, -0.4), (2, -0.9)\}$ define the training set. The error surface is shown in Figure 2.

A perceptron with linear activation function has an elliptical error surface centered at the full least squares estimates. However, the non-linear activation function turns the surface irregular but with a distinct minimum at $w_o = -0.207$ and $b_o = -1.045$. Both norm and sum of the absolute weights functions have their minimum at the origin ($w = 0$, $b = 0$). The associated Pareto sets for norm and absolute weights functions are sets of solutions that start from origin and end at the minimum error point. To compare the solutions conditioned to the previous constraints, the error contours as well as the norm and the LASSO contours are shown in Figure 3. The constraint region for the norm is the disk $w^2 + b^2 \le \eta$ while that for LASSO is the diamond $|w| + |b| \le t$. Both methods find the first point where the error contours hit the constraint regions which represents a solution with minimum error conditioned to the respective constraint. Unlike the disk, the diamond has corners, if the solution occurs at a corner then it has one weight equal to zero. When the number of weights is larger than 2, the diamond becomes a rhomboid, and has many corners, flat edges and faces, there are many more opportunities for the estimated weights to be zero [14].

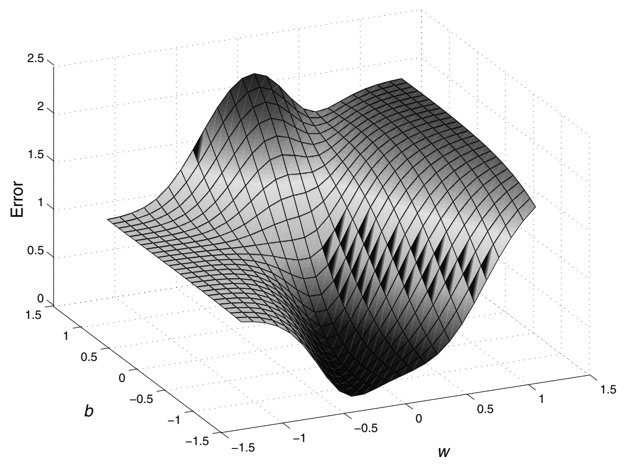


Fig. 2. Perceptron's Error Surface for the training set: $(x_i, y_i) = \{(-3, -0.4), (2, -0.9)\}$.

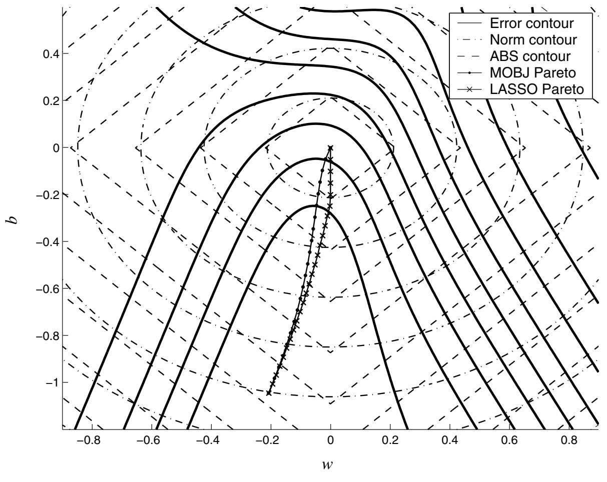


Fig. 3. Error, Norm and LASSO contours.

Although MOBJ and LASSO methods have common solutions the path between origin and minimum error are quite different. Figure 3 shows that the norm solutions are non-zero for any constraint except at the origin. The LASSO approach have a sub-set of solutions where w is null. As the solutions reach the minimum error, they become closer. The main advantages of the LASSO constraint are the capability to control generalization just as the norm constraint does but with an automatic weight selection procedure.

For multi-layer perceptron (MLP) neural networks as long as the topology is larger than the unknown optimal number of hidden nodes, the LASSO approach can reduce or eliminate some weights and network inputs on the final solution. This may result on a reduced topology and improved network performance. Moreover, this information can be used as a data underlying function complexity measure.

IV. Methodology

The LASSO method for neural networks can be implemented with the *ellipsoidal algorithm* [9], the same algorithm used by MOBJ [8] to search the norm constrained solutions. First, the multi-objective problem can be written as a constrained mono-objective optimization:

$$\mathbf{w}^* = arg \min_{w} J(w) \qquad subject\,to: \; g(w) \leq 0 \tag{3}$$

where $w \in \Re^p$, $J(w) = \sum_{j=1}^{N} \left(y_j - y(\mathbf{w}, \mathbf{x}_j)\right)^2$, $g(w) = \left(\sum_j |w_j|\right) - \varepsilon_i$, ε_i is the limit of equality for the sum of the absolute weights' function $\left(\sum_j |w_j| \leq \varepsilon\right)$ and $y(x_i, w)$ is the neural network response.

The conventional ellipsoidal method can be written as:

$$m(w) = \begin{cases} \nabla g(w), \text{ if } g(w) > 0 \\ \nabla J(w), \text{ if } g(w) \leq 0 \end{cases} \tag{4}$$

where $\nabla(.)$ means the gradient or any subgradient of the argument. The weight update equation is:

$$w_{k+1} = w_k - \beta_1 \frac{Q_k m_k}{\left(m_k^T Q_k m_k\right)^{1/2}} \tag{5}$$

where:

$$Q_{k+1} = \beta_2 \frac{Q_k - \beta_3 (Q_k m_k)(Q_k m_k)^T}{m_k^T Q_k m_k} \tag{6}$$

where w_0 is a random initial vector and Q_0 is a symetric positive matrix, in general, the identity matrix, $\beta_1 = \frac{1}{N+1}$, $\beta_2 = \frac{N^2}{N^2-1}$ and $\beta_3 = \frac{2}{N+1}$.

A subgroup of efficient solutions can be generated from a sequence of normally equidistant constraints $\varepsilon_1, \varepsilon_2, ..., \varepsilon_{max}$ such that, $\varepsilon_1 < \varepsilon_2 < ... < \varepsilon_{max}$, where for each constraint, the ellipsoidal method is used to generate a single solution being the final solution selected as the one that has the least validation error.

The algorithm needs the gradient vector of the error and the sum of absolute weights. The following equations were used to calculate the absolute weight function derivative:

$$|w_j| = \begin{cases} +w_j, \text{ if } w_j \geq 0 \\ -w_j, \text{ if } w_j < 0 \end{cases} \tag{7}$$

$$\frac{\partial |w_j|}{\partial w_j} = \begin{cases} +1, \text{ if } w_j \geq 0 \\ -1, \text{ if } w_j < 0 \end{cases} \tag{8}$$

As described in Equation 8 the absolute weight derivative is discontinuous at the origin. In addition to this, the ellipsoidal algorithm performs an iterative search. Consequently, the generated solutions will not achieve some exactly null weights but it will generate some weights with very small amplitudes, closer to zero. In order to detect and eliminate those weights we use a very simple procedure that was previous proposed to simplify MOBJ solutions [16]. We randomly select weights in the final neural network solution, if the validation error decreases when the weight is set to zero then it is definitely extracted, otherwise it is restored and a different weight is randomly selected. The procedure is repeated until every weight in the network has been tested. This approach provides a fare comparison of the effective number of weights between MOBJ and LASSO solutions.

Four data sets were chosen in order to compare and evaluate MOBJ, LASSO and Early-Stopping solutions. The algorithms were set with the same initial random weights and the training stop was based on validation criterion. The Data sets were divided into training, validation and test sets with 50%, 25% and 25%, respectively and 30 simulations were performed for each algorithm. For each simulation the data sets were randomly divided and at the end of the training process the networks were pruned with the random selection procedure.

The first data set was generated from the following function:

$$f(x) = \frac{(x-2)(2x+1)}{1+x^2} + \epsilon \tag{9}$$

where x were generated form a uniform distribution within the range $[-8; 12]$ and ϵ is a random variable normally distributed with 0 mean and standard deviation $\sigma = 0.2$. The data set consists of 200 samples. The neural network topology used has one input, one output, 15 hidden nodes enumerating 46 weights. The number of nodes was chosen empirically.

The next data sets used are available at `http://www.ics.uci.edu/~mlearn/MLRepository.html` (2001). The Boston data set [17] represents a regression problem where the output is the median value of owner-occupied homes in $1000's and the inputs are 13 variables related to physical characteristics of the house and neighborhood quality. It has 506 instances. The Cancer data set consists of 9 input attributes, 2 possible outputs and 699 instances. The inputs are numerical attributes of cancer cells. The outputs associate each input to one of the two following classes: benign or malignant (tumor). The Card data set is concerned with credit card applications, with 51 input attributes, 2 possible outputs and 690 instances. The inputs correspond to a mix of continuous and nominal attributes and the outputs associates the acceptance or not for personal credit. Both Cancer and Card data sets represent classification problems. Similarly to the function regression problem, all networks were created with 15 hidden nodes. The respective numbers of weights in the networks for each problem are: 226 for Boston, 151 for Cancer and 781 for Card.

V. Results

The main difference between MOBJ and LASSO constrained solutions is demonstrated in Figure 4 for the Boston data set In this case, no pruning was performed. The MOBJ approach reduces the norm of the weights and consequently theirs amplitudes but it does not aggregate the functionality of selecting weights as demonstrated by LASSO. As described in Section III: "the LASSO solutions have many more opportunities for the estimated weights to be zero."

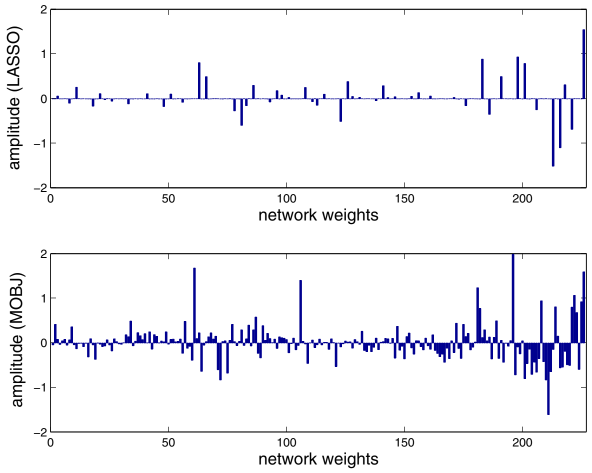


Fig. 4. Amplitude of the weights for Boston data set trained with MOBJ and LASSO.

Tables I, II and III present some descriptive statistics (Stat.): mean (ave), standard deviation (std) and trained network solutions with minimum number of weights (w^*) for each data set. For $f(x)$ and Boston data sets, *Train.*, *Valid.* and *Test* represent training error (residual sum of squares), validation error and test error. For Cancer and Card data sets they represent the percentage of correct classification patterns for training, validation and test sets. The norm of the weights ($||\mathbf{w}||$) and the sum of the absolute weights ($\sum_j |w_j|$) are also included. n_w represents the number weights after pruning. CPU$_{time}$ describes the training time (in seconds) on a Pentium IV, 2.4GHz, 1GB (RAM) using the software MATLAB. *Train.** *Valid.** and *Test** are the error results after pruning.

Figure 5 shows the approximations performed by MOBJ, LASSO and Early-Stopping for the non-linear regression problem ($f(x)$). In general both methods (MOBJ and LASSO) were able to provide good solutions and on average both mean squared error for training (*Train.*) and validation (*Valid.*) sets are close to the known error variance ($\sigma^2 = 0.04$) which emphasizes that the solutions have achieved good generalization. The Early-Stopping method presented, on average a mean training error close to $\sigma^2 = 0.04$ but with higher validation and test errors. The LASSO best solution is a network with 16 weights that represents a fully connected network with 5 hidden nodes against 27 weights found by MOBJ that is associated to a 10 hidden nodes network and 28 weights found by Early-Stopping.

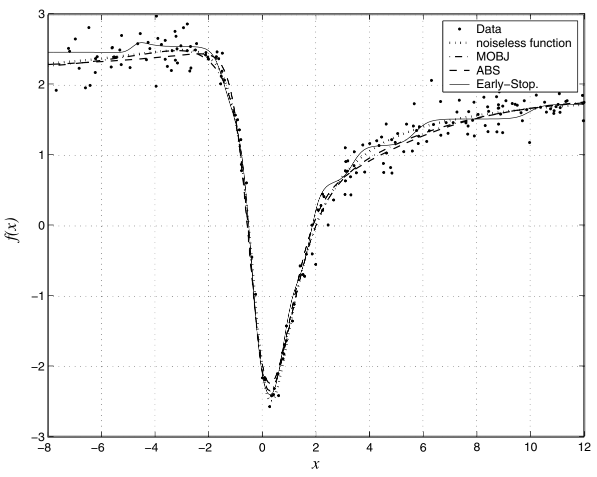


Fig. 5. Approximated function with MOBJ, LASSO and Early-Stopping.

In general, MOBJ, LASSO and Early-Stopping achieved solutions with closer errors and classification rates for the training set. For validation and test sets, MOBJ and LASSO methods obtained the best results. As expected, LASSO solutions reached least sum of the absolute weights and least number of parameters after pruning. MOBJ solutions achieved least norm function values but very close to the LASSO ones. On the contrary, LASSO solutions have lower sum of the absolute weights in comparison to MOBJ. The Early-Stopping solutions present the highest values of norm and sum of the absolute weights and also increased error for the test set. However, it presented the least computational time followed by MOBJ and LASSO algorithms. Methods MOBJ and LASSO generate multiple solutions being the final one selected based on validation error. The Early-Stopping algorithm only generates a single solution. This factor must be considered when evaluating processing time of each method.

For the $f(x)$ function, the Early-Stopping algorithm achieved the highest error for the test set. LASSO method provided the least training, validation and test errors for Boston data set after pruning. For CANCER and CARD database, the LASSO method presented slightly superior classification rates when compared to MOBJ and Early-Stopping results.

The random pruning technique provided a remarkable simplification of the networks. The number of extracted weights were 30 weights for $f(x)$ (65.2%), 204 weights for Boston (90.27%), 145 weights for Cancer (96.03%) and 774 weights for Card (99.1%). For some cases like Cancer and Card the solutions with minimum number of weights were also generated by MOBJ. However, on average, LASSO provided solutions with both minimum number of weights and standard deviation. It is important to notice that the random pruning technique is able to generate MOBJ solutions with reduced number of weights but this feature is not as frequent as for LASSO solutions.

Based on validation and training criterion, the network found by MOBJ with minimum training and validation errors has 14 hidden nodes against 6 hidden nodes found by LASSO technique for Boston data set. Figure 6 shows the simplified topology found with best generalization performance. In this case: $E_{train} = 0.1324$, $E_{val} = 0.1418$ and $E_{test} = 0.1198$ are the respective training, validation and test errors for MOBJ and $E_{train} = 0.0525$, $E_{val} = 0.1340$ and $E_{test} = 0.1156$ for LASSO. Although it was not evident, the method's capability to exclude inputs, only one input has been eliminated from the trained network with LASSO. Figure 7 emphasizes the input selection property of LASSO. From 51 initial inputs only 13 were left with the LASSO algorithm against 16 with MOBJ for Cancer data set. Classification rates are 97.43% (train.), 97.71% (valid.) and 96.55% (test.) for MOBJ and 97.43% (train.), 97.71% (val.) and 98.85% (test.) for LASSO. Quite the same classification rates with a slight positive difference for LASSO. Early-Stopping results were not considered here because they presented smaller classification rates.

VI. Conclusions

This paper presented the LASSO multi-objective algorithm. The algorithm is able to reach high generalization solutions for regression and classification problems. In addition, the sum of the absolute weights objective function aggregates topology selection into the training process. The method was compared with the original Multi-Objective algorithm and Early-Stopping and it achieved results with high generalization responses and reduced topology. To measure the effective number of weights for each algorithm a random selection procedure were applied to both network solutions. Although the pruning approach is also able to efficiently simplify original MOBJ and Early-Stopping solutions, it performed better with LASSO trained networks. Results show that the LASSO method is able to eliminate inputs and provide topology selection what makes it an effective neural network data mining tool.

Acknowledgment

This work was supported by FAPEMIG.

References

[1] D. Rumelhart, G. Hinton and R. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[2] S. Haykin, *Neural networks: A comprehensive foundation*, MacMillan: New York, 1994.

[3] D. Rumelhart and J. McClelland, *Parallel Distributed Processing*, vol. 1 & 2, MIT Press:Cambridge, MA, 1986.

[4] P. L. Bartlett, "For valid generalization, the size of the weights is more important than the size of the network," in *Proceedings of NIPS 9*, pp. 134–140, 1997.

[5] L. Prechelt, "Automatic early stopping using cross validation: quantifying the criteria," *Neural Networks*, vol. 11(4), pp. 761–767, 1998.

[6] A. Krogh and J. A. Hertz, "A Simple Weight Decay Can Improve Generalization," in *Proceedings of NIPS*, vol. 4, pp. 950–957, 1991.

[7] R. Reed. Pruning algorithms - a survey. *IEEE Transactions on Neural Networks*, 4(5):740746, 1993.

[8] R. A. Teixeira, A. P. Braga, R. H. C. Takahashi and R. R. Saldanha, "Recent Advances in the MOBJ Algorithm for training Artificial Neural Networls," *International Journal of Neural Systems*, vol. 11, pp. 265–270, 2001.

[9] R. G. Bland, D. Goldfarb and M. J. Todd, "The ellipsoid method: a survey," *Operations Research*, vol. 29(6), pp. 1039–1091, 1981.

[10] M. A. Costa, A. P. Braga, B. R. Menezes, R. A. Teixeira and G. G. Parma, "Training neural networls with a multi-objective sliding mode control algorithm," *Neurocomputing*, vol. 51, pp. 467–473, 2003.

[11] M. A. Costa, A. P. Braga and B. R. Menezes, "Improved Generalization Learning with Sliding Mode Control and the Levenberg-Marquadt Algorithm," in *Proceedings of VII Brazilian Symposium on Neural Networks*, 2002.

[12] Y.Jin, T. Okabe and B.Sendhoff, "Neural network regularization and ensembling using multi-objective algorithms," *In: Proceedings of the 2004 IEEE Congress on Evolutionary Computation*, pp. 1-8, 2004.

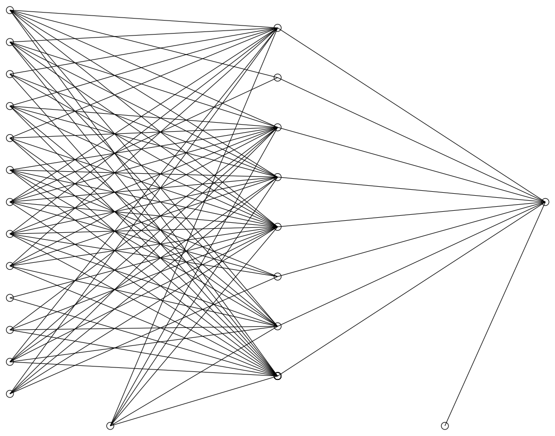
[13] J.E. Fieldsend and S. Singh, "Optimizing forecast model complexity using multi-objective evolutionary algorithms," *Applications of Multi-Objective Evolutionary Algorithms*, World Scientific, pp. 675-700, 2004.

[14] T. Hastie, R. Tibshirani and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, Springer Series in Statistics, 2001.

[15] R. Tibshirani, " Regression Shrinkage and Selection via the Lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58(1), pp. 267-288, 1996.

[16] M. A. Costa, A. P. Braga and B. R. Menezes, "Improving neural networks generalization with new constructive and pruning methods," *Journal of intelligent & Fuzzy Systems*, vol. 13, pp. 73–83, 2003.

[17] D. Harrison and D. L. Rubinfeld, "Hedonic prices and the demand for clean air," *J. of Eviron. Economics & Management* , vol. 5, pp. 81–102, 1978.


(a) MOBJ solution

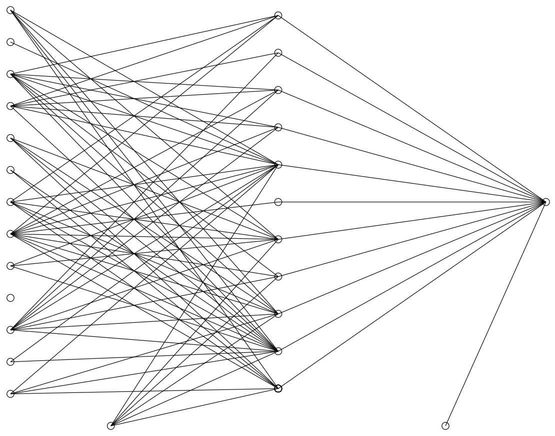
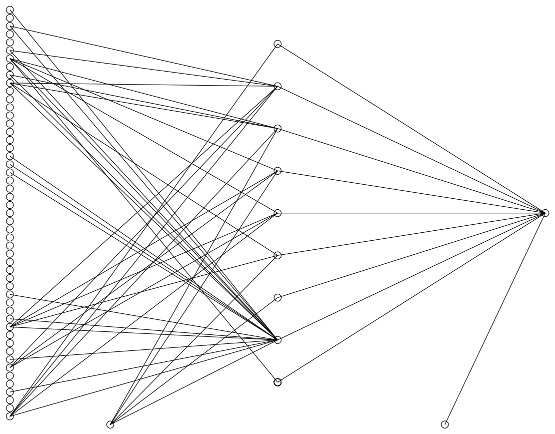

(b) LASSO solution

Fig. 6. Pruned Networks for Boston data set.


(a) MOBJ solution

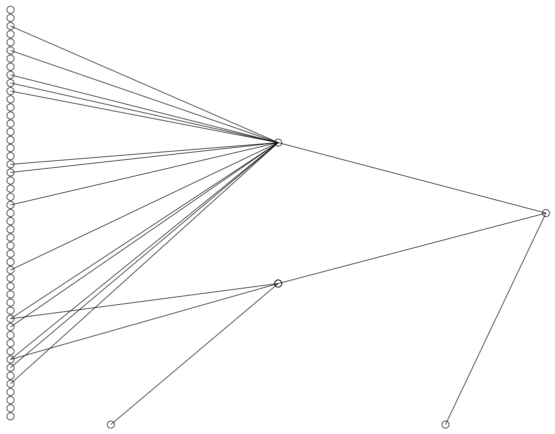

(b) LASSO solution

Fig. 7. Pruned Networks for Cancer data set.

TABLE I
MOBJ RESULTS

Data	Stat.	Train.	Valid.	Test	$\|\|\mathbf{w}\|\|$	$\Sigma_j\|w_j\|$	n_w	CPU$_{time}$	Train.*	Valid.*	Test$*$
	w^*	0.0335	0.0490	0.0507	6.01	22.39	27	47.77	0.0334	0.0490	0.0508
$f(x)$	ave	0.0398	0.0404	0.0471	5.87	23.61	36	39.52	0.0406	0.0403	0.0482
	std	0.0062	0.0086	0.0112	1.10	4.87	8	13.53	0.0068	0.0086	0.0113
	w^*	0.1654	0.1852	0.1886	2.31	8.90	28	77.33	0.2889	0.1806	0.1914
Boston	ave	0.0865	0.1578	0.1970	4.44	30.15	75	88.32	0.1665	0.1524	0.2303
	std	0.0665	0.0462	0.0597	1.60	16.62	36	16.93	0.0886	0.0469	0.0603
	w^*	0.9800	0.9486	0.9828	1.49	3.24	6	28.69	0.9714	0.9543	0.9770
Cancer	ave	0.9781	0.9741	0.9575	3.53	18.22	55	33.76	0.9670	0.9758	0.9546
	std	0.0093	0.0121	0.0184	2.33	11.44	41	8.26	0.0119	0.0100	0.0176
	w^*	0.9246	0.8728	0.8488	2.04	4.04	7	354.58	0.8638	0.8902	0.8430
Card	ave	0.8923	0.8798	0.8543	1.89	8.40	62	346.57	0.8562	0.8815	0.8463
	std	0.0264	0.0163	0.0267	1.71	6.46	48	55.98	0.0181	0.0178	0.0220

TABLE II
LASSO RESULTS

Data	Stat.	Train.	Valid.	Test	$\|\|\mathbf{w}\|\|$	$\Sigma_j\|w_j\|$	n_w	CPU$_{time}$	Train.*	Valid.*	Test$*$
	w^*	0.0406	0.0364	0.0411	9.46	26.17	16	326.11	0.0405	0.0360	0.0406
$f(x)$	ave	0.0416	0.0399	0.0481	6.38	19.52	26	165.11	0.0425	0.0397	0.0490
	std	0.0071	0.0080	0.0094	1.18	4.32	9	109.76	0.0084	0.0080	0.0100
	w^*	0.1075	0.1137	0.3761	2.53	8.08	22	187.95	0.1123	0.1135	0.3736
Boston	ave	0.0924	0.1532	0.1993	4.39	20.23	37	160.42	0.1358	0.1466	0.2130
	std	0.0438	0.0433	0.0740	1.33	9.32	15	17.20	0.0465	0.0409	0.0646
	w^*	0.9829	0.9600	0.9598	5.16	11.49	6	68.73	0.9686	0.9657	0.9540
Cancer	ave	0.9741	0.9758	0.9625	5.49	13.78	10	76.56	0.9704	0.9768	0.9602
	std	0.0087	0.0118	0.0138	4.34	11.44	3	17.99	0.0100	0.0110	0.0142
	w^*	0.8899	0.8960	0.8430	2.49	4.81	7	330.61	0.8551	0.8960	0.8430
Card	ave	0.8781	0.8802	0.8539	2.04	4.83	13	304.18	0.8589	0.8819	0.8473
	std	0.0237	0.0174	0.0201	1.30	3.41	4	42.29	0.0155	0.0177	0.0194

TABLE III
EARLY STOPPING RESULTS

Data	Stat.	Train.	Valid.	Test	$\|\|\mathbf{w}\|\|$	$\Sigma_j\|w_j\|$	n_w	CPU$_{time}$	Train.*	Valid.*	Test$*$
	w^*	0.0323	0.0741	0.0469	53.90	214.54	28	237.41	0.0453	0.0656	0.0391
$f(x)$	ave	0.0370	0.0565	0.0586	54.72	219.25	39	57.52	0.0508	0.0522	0.0667
	std	0.0061	0.0118	0.0179	1.06	3.14	4	70.38	0.0253	0.0085	0.0252
	w^*	0.3216	0.2237	0.2851	13.18	112.68	27	1.33	0.4945	0.1982	0.4065
Boston	ave	0.0877	0.1607	0.1886	12.37	108.66	76	52.63	0.1876	0.1488	0.2346
	std	0.0517	0.0452	0.0592	0.84	5.63	21	47.51	0.0758	0.0434	0.0804
	w^*	0.9886	0.9600	0.9310	19.63	203.74	6	49.14	0.9600	0.9657	0.9253
Cancer	ave	0.9741	0.9690	0.9630	19.53	205.19	16	45.27	0.9591	0.9726	0.9588
	std	0.0092	0.0137	0.0150	1.04	7.02	9	46.21	0.0089	0.0126	0.0168
	w^*	0.9014	0.8671	0.8198	3386.84	23437.99	3	12.47	0.8522	0.8902	0.8256
Card	ave	0.8977	0.8516	0.8498	2875.94	17936.89	15	21.65	0.8523	0.8697	0.8469
	std	0.0192	0.0219	0.0250	514.93	6059.30	10	15.02	0.0202	0.0179	0.0220