Variable Selection with Neural Networks

Tautvydas Cibas *, Françoise Fogelman Soulié **, Patrick Gallinari °, Sarunas Raudys#

* LRI, bât. 490, Université de Paris-Sud, F-91405, Orsay, (France).
** SLIGOS,Tour Anjou, 33 quai de Dion Bouton, F- 92814 Puteaux cedex (France)
° LAFORIA-IBP, Univ. Paris 6, 4 place Jussieu, F-75232 Paris cedex 05, (France).
Dep. Data Analysis, Inst.Math.Informatics, Akademijos 4,Vilnius 2600, (Lithuania).

1. Introduction

Neural Networks -NN- are used in quite a variety of real-world applications, where one can usually measure a potentially large number N of variables X_i; probably not all X_i are equally informative: some should even be considered as noise to be eliminated. If one could select $n << N$ "best" variables X_i, then one could reduce the amount of data to gather and process -hence costs- while possibly increasing performances. *Variable selection* is thus an important issue in NNs. It is also a complex problem; one needs a criterion to measure the value of a variables subset and that value will of course depend on the predictor or classifier further used: a subset of variables could be optimal for one classifier, and very inefficient for another; a large subset might not contain all variables of a smaller subset (non-*monotonicity*). Conventional variable selection techniques are based upon statistical or heuristics tools [Fukunaga, 90]: the major difficulty comes from the intrinsic combinatorics of the problem. Using NNs for variable selection is attractive since one can globally adapt the variables selector together with the classifier. We have described*OCD* [Cibas *et al.*, 94], a selection technique based on an extension of weight pruning. As was shown by various authors recently [McKay, 92b], pruning is a heuristic method to constrain the classifier. More generally, regularization terms can achieve the same goal: by adding to the cost function a term which embodies the prior on the solution, enforce desired properties: for example one could retain only large weights (*weight decay* [Rumelhart *et al.*, 86], [Hanson *et al.*, 89], [Weigend *et al.*, 91]) or get a gaussian mixture [Nowlan *et al.*, 92]. We present in this paper a regularization approach to variable selection, where the regularization term (gaussian and gaussian mixture priors) allows to discard, during training, least useful variables, and we compare performances to the `stepwise` procedure. When implementing these different methods, one has to decide when to start (pruning, or regularizing) and how much (how many weights, or with how large a regularizing factor). Techniques such as OCD require that the optimum be reached. However, extensive evidence [Finnoff *et al.*, 93] shows that it is more efficient to use "non convergent methods", where modification is started before full convergence is reached. We thus also compare our regularization approach to the pruning technique described in [Cibas *et al.*, 94]. We illustrate our method on two relatively small problems: prediction of a synthetic time series and classification of waveforms [Breiman *et al.*, 84]. The paper is organized as follows: section 2 introduces notations and results from the literature; section 3 problems used to test our methods and section 4 variable selection by regularization.

2. Variable Selection

2.1. Definitions

Let a random variable pair $(X, Y) \in \Re^N \times \Re^p$ be given, with probability distribution P. Based on a sample $D_m = \{(x^1,y^1)\ldots(x^m,y^m)\}$, drawn from (X, Y), we train a NN α, and produce an estimator F, which depends upon α and D_m. Different nets may have different performances: here we select nets depending upon their respective empirical errors. In this paper, we chiefly compare nets differing only in their input dimension: some components of the original input $X \in \Re^N$ are eliminated to produce a vector $x \in$

$\Re^n$, with $n \leq N$. The problem in variable selection is to extract the best set of features, and possibly determine the optimal number n* of features, for a given criterion.

2.2. Statistical methods for variable selection/extraction

In the following, we compare our NN methods to a conventional technique (Fisher-Snedecor ratio) implemented in [SAS], a F-test of the null hypothesis H_0 that all selected variables are significant. There are various versions of the technique: in the *sequential forward* method, vector x is progressively built up, by adding one feature at a time; the *sequential backward* technique works in opposite direction: features are progressively eliminated from X one at a time; the *stepwise* procedure alternates both directions. However, results of variable selection using these different procedures tend to be rather unstable in the resulting variable subset.

2.3. Neural Networks and regularization

The regularization framework provides methods to handle the problem of poor generalization. The idea is that estimation with too few and noisy data is an ill-posed problem, so that constraints must be imposed on the final solution to make the problem well-posed. These constraints are usually implemented as follows.

Let N be a NN picked from a family of multi-layer networks with n inputs. For an input x^k, desired output y^k and computed output $F(x^k)$, we train the network to minimize:

$$M(W) = \alpha\, C(W, D_m) + \beta\, E_W(W) \qquad (2\text{-}1)$$

where $C(W, D_m) = \frac{1}{2m} \sum_{k=1\ldots m} \| y^k - F(x^k) \|^2 \qquad (2\text{-}2)$

is the Mean Squared Error -MSE- and the *regularization term* $E_W(W)$ embodies the desired constraints. Weight decay -WD- is the most commonly used regularization term [Rumelhart *et al.*, 86], [Weigend *et al.*, 91], [Chauvin, 89]:

$$E_W(W) = \sum_j w_j^2 \qquad (2\text{-}3)$$

In the Bayesian framework [McKay, 1992a, 1992b], one looks for weights W which maximize the a posteriori distribution, conditionned by the observed data D_m; which is equivalent, if one assumes an additive Gaussian noise model $N(0, \sigma^2)$, to minimizing:

$$M(W) = \frac{1}{2\sigma^2} C(W, D_m) - ln\ P(W) \qquad (2\text{-}4)$$

This is (2-1), with the additional interpretation that the regularization term is the log of the weights prior. If this prior is gaussian $N(0, \sigma_W^2)$, then the log prior is, up to a constant, just weight decay (2-3) (where $|W|$ is the number of weights):

$$ln\ P(W) = -|W|\ ln\ \sigma_W \sqrt{2\pi} - \frac{1}{2\sigma_W^2} \sum_j w_j^2 \qquad (2\text{-}5)$$

Other assumptions on the weights prior can be made: for example, a gaussian mixture [Nowlan *et al.*, 92]. This Bayesian framework can be used for pruning [McKay, 93], [Thodberg, 93]. The regularization factors α and β have an interpretation in terms of the noise and weight variances. In the standard regularization approaches, α and β have to be determined by cross validation, whereas the full Bayesian approach can provide an automatic determination of all parameters W, α, β. In the following, we use a regularized cost function to perform feature selection, with 2 different weights priors: gaussian -WD- and gaussian mixture.

3. Test problems

In this section, we describe the version of Gradient Back Propagation -GBP- and testbed problems we used (see [Cibas *et al.* 94] for more details).

3.1. Training procedure

Our training procedure is an on-line GBP implemented as follows: the gradient step ε is decreased as soon as the relative average cost variation becomes too small.

GBP • run N steps of on-line GBP with $\varepsilon = \varepsilon_0$. $\varepsilon(N+1) = \varepsilon_0$. (3-1)

• while $\varepsilon \geq \varepsilon_1$ do

run on-line GBP with ε

if $r(W, D_m^v, t, T_1, 1, T_1) \leq \theta_1$

then $\varepsilon(t+1) = \alpha \cdot \varepsilon(t)$ else $\varepsilon(t+1) = \varepsilon(t)$

endwhile

$$[C(W,D)]_t^{T_1} = \frac{1}{T_1} \sum_{\tau=0}^{T_1-1} C(W, t-\tau, D) \quad , \quad r(W,D,t,T_1,T_2,T_3) = 1 - \frac{[C(W,D)]_{t+T_2}^{T_3}}{[C(W,D)]_t^{T_1}} \quad (3\text{-}2)$$

GBP thus depends on various parameters: ε_0, ε_1 and θ_1 being given, N, T_1 and α are set by cross-validation, using MSE for time series, performance for waveforms, on a validation set.

3.2. Time Series

We use an artificial time series y_t, where ε_t is a white noise, to train a network 10-7-1 (with all 10 input variables x_t, 7 hidden units and 1 linear output):

$$y_t = 0.3\, x_{t-6} - 0.6\, x_{t-4} + 0.5\, x_{t-1} + 0.3\, x_{t-6}^2 - 0.2\, x_{t-4}^2 + \varepsilon_t \quad (3\text{-}3)$$

Three independent samples of size m=1000 each are drawn from series y_t (D_m^l for learning, D_m^v validation and D_m^t test.) Cross validation gave T_1 = 5, N = 30, and α = 3/4 (for ε_0 = 0.1, ε_1 = 0.0001, θ_1 = 0.0005). Final MSEs at the end of GBP were: 0.04359 on D_m^l, 0.5050 on D_m^v and 0.04672 on D_m^t.

3.3. Waveforms

Three vectors or *waveforms* [Breiman *et al.*, 84] in 21 dimensions, H^i, i=1,…,3, are defined. Two by two noisy convex combinations are formed [de Bollivier *et al.*, 91] and the problem is to classify in one of the 3 classes corresponding to the 3 pairs (m,n):

$$X_i = \frac{1}{5}[\, u\, H_i^m + (1-u)\, H_i^n \,] + \varepsilon_i, \; 0 \leq i \leq 20; \qquad X_i = \varepsilon_i, \qquad 21 \leq i \leq 40 \quad (3\text{-}4)$$

where ε_i is a white noise. D_m^l has 300 elements, D_m^v 1000, D_m^t 5000. On a network 40-30-20-3 (2 hidden layers of 30 and 20 neurons), cross-validation gave: T_1 = 5, N = 15 and α = 15/16 (for ε_0 = 0.25 ε_1 = 0.001, θ_1 =0.0001). Final performances of GBP on learning, validation and test were 99.67, 83.80 and 80.88 respectively.

4- Regularization

4.1. Introduction

We will restrict our analysis here to the case of a multilayer perceptron (MLP) with only one hidden layer, and full connections. Extension to more complex architectures is straightforward. Our technique is somewhat similar to [Nowlan *et al.*, 92]. Let us suppose that our MLP has N inputs, h hidden units and p output units, and denote V, resp. W, the weights from input to hidden layers, resp. hidden to output layer. V_i is thus the weight vector exiting from unit i: discarding feature i is equivalent to setting V_i to 0. With these notations, equation (2-1) becomes:

$$M(V, W) = \alpha \frac{1}{2m} \sum_{k=1\ldots m} \| y_k - F(x_k) \|^2 - \beta \ln P(V, W) \quad (4\text{-}1)$$

We will use two different priors P(V, W): a gaussian and a gaussian mixture.

4.2. Weight decay

To select input variables, we will take for $ln\ P(V, W)$ (which is not plain WD (2-5)):

$$ln\ P(V, W) = - \sum_{i=0}^{N} \sum_{j=1}^{h} V_{ji}^2 \qquad (4\text{-}2)$$

We normalize and take $\lambda = \beta / \alpha$. Our algorithm is then as follows:

Regularization (4-3)

- run N steps of on-line GBP with $\varepsilon = \varepsilon_0$ and cost function C.
- set $\varepsilon(N+1) = \varepsilon_0$ and from now on use cost function M.
- while $\varepsilon \geq \varepsilon_1$ do

  run on-line GBP with ε

  if $r(W, D_m^v, t, T_1, 1, T_1) \leq \theta_1$ then $\varepsilon(t+1) = \alpha \cdot \varepsilon(t)$ else $\varepsilon(t+1) = \varepsilon(t)$

endwhile

where r is defined as before by (3-2), i.e. using function C.

This algorithm is executed, with the values of N, T_1 and α as set by cross-validation after GBP (see 3.1.), for the given values of ε_0, ε_1 and θ_1. Cross validation is used to set parameter λ^*. Let us then define, for every input variable i, its weight variance var_i.

Variable selection is then run along (4-4) and q set by cross validation (for λ^*):

Variable selection (4-4)

- order variances: $var_{i_1} \geq var_{i_2} \geq \ldots \geq var_{i_N}$
- choose n such that: $\sum_{k=1}^{n} var_{i_k} \geq q \sum_{k=1}^{N} var_{i_k}$
- eliminate variables i_{n+1} to i_N
- run N' steps of on-line GBP with $\varepsilon = \varepsilon_1$ and cost function M.

Results are shown in figure 1 for time series and waveforms and compared to performances obtained by the network retrained on the selected sets of variables (fig. 2) as only inputs. The set selected by SAS (stepwise) is also given for comparison. These results show that our WD-based selection worked well for the time series (it allows to reach a better MSE with the selected variables: net 4-7-1, than SAS, with net 3-7-1. It is also significantly better than net 10-7-1. For waveforms, the selected net 20-30-20-3 also achieves better performances on test set; in fig. 1 and 2, the selected sets are denoted 14* (with variables 7,11,17,12,5,19,18,13,4,10,29,38,39,40); 17*, 19*, 20* and 21* include the 17, 19 … first variables in the list: 12,13,17,7,11,6, 18,4,19,14,5,9,10,15,16,3,20,39,31,21,33: notice that the variables selected by SAS are fewer and come more from the "noise" part of the signal.

4.3. Gaussian mixture prior

Assume a priori that weights V and W are independent (not true any more as soon as learning starts!) and that W, resp. V_0 is Gaussian $N(0, \sigma_W^2)$, resp. $N(0, \sigma_l^2)$. V_i's are supposed to be independently distributed with a Gaussian mixture of two components: one -in proportion π- $N(0, \sigma_l^2)$ has a large variance and the other $N(0, \sigma_s^2)$ a small one. Under these assumptions, we have:

$$ln\ P(V, W) = \frac{1}{2\sigma_W^2} \sum_{s=1}^{p} \sum_{j=0}^{h} W_{sj}^2 + \frac{1}{2\sigma_l^2} \sum_{j=1}^{h} V_{j0}^2$$

$$- \sum_{i=1}^{N} ln\left[\frac{\pi}{\sigma_s^{2h}} e^{-\frac{1}{2\sigma_s^2}\sum_{j=1}^{h} V_{ji}^2} + \frac{1-\pi}{\sigma_l^{2h}} e^{-\frac{1}{2\sigma_l^2}\sum_{j=1}^{h} V_{ji}^2} \right] \quad (4\text{-}3)$$

$M(V, W)$ is minimized, through an on-line GBP for example (3-1). In the following, all regularization parameters are held constant ($\sigma_W{}^2=\sigma_l{}^2=1$, $\sigma_s{}^2=0.005$), while weights V and W are updated; π, σ^2 are set by cross validation on Regularization: only results with the best σ^2 are given. Results for time series are shown in figure 3. Comparison of performances of the nets with selected variables (fig. 2) shows that, finally, it is the set of variables selected by this Gaussian mixture regularization which gives the best performances on test set: notice that that technique has learnt to use both "real" variables 10,7,5 (x_{t-1},x_{t-4},x_{t-6}) and "hidden" variables 9,(6),4 coming from y_{t-1}.

5. Conclusion

We have presented here a technique based on regularization for variable selection, with two different regularizing terms, coming from different hypothesis on the weights priors (Gaussian: weight decay and mixture of Gaussians). The regularization with gaussian mixture gave the best final performances on test set for a synthetic time series example, as compared to the WD-based method, a conventional statistic method (`stepwise` in SAS) or a pruning method [Cibas *et al.*, 94]. However, our method still suffers from various problems:

- there are many parameters which have to be set by cross validation. This is far too computation intensive in practical problems. In particular, our version of GBP should be replaced by a parameter-free technique, such as e.g. conjugate gradients [Møller, 93].
- the results are very sensitive to the a priori choice of parameters (e.g. $\sigma_W{}^2$, $\sigma_l{}^2$, $\sigma_s{}^2$).

We are presently working on an improved version of our technique where these parameters are adapted in the course of learning, through the same GBP technique.

6. References

BOLLIVIER de M., GALLINARI P., THIRIA S.: Cooperation of neural nets and task decomposition. IJCNN'91, Seattle, vol. II, 573-576, (1991).

BREIMAN L., FREIDMAN J., OLSHEN R., STONE C.: Classification and regression trees. Wadsworth Int. Group. (1984).

CHAUVIN Y.: A back-propagation algorithm with optimal use of hidden units. In "Neural Information Processing Systems", NIPS'88, D. Touretzky ed., vol. 1, Morgan Kaufmann, 519-526. (1989).

CIBAS T., FOGELMAN SOULIE F., GALLINARI P., RAUDYS S.: Variable Selection with Optimal Cell Damage. ICANN'94, Springer Verlag. (1994).

FINNOFF W., HERGERT F., ZIMMERMANN H.G.: Improving model selection by nonconvergent methods. Neural Networks, vol. 6, n° 6, 771-783, (1993).

FUKUNAGA K.: Statistical Pattern Recognition. Academic Press, 2nd edition. (1990).

HANSON S.J., PRATT L.Y.: Comparing biases for minimal network construction with back-propagation. In "Neural Information Processing Systems", NIPS'89, D.S. Touretzky ed., Morgan Kaumann, vol. 1, 177-185, (1989).

McKAY D.J.C.: Bayesian interpolation. Neural Computation, vol. 4, 415-447, (1992a).

McKAY D.J.C.: A practical bayesian framework for backpropagation networks. Neural Computation, vol. 4, 448-472, (1992b).

McKAY D.J.C.: Bayesian non-linear modeling for the energy prediction competition. Tech. Rep., University of Cambridge, (1993).

MOLLER M.: Efficient training of feed-forward neural networks. Thesis, Univ. Aarhus, (1993).

NOWLAN S.J., HINTON G.E.: Simplifying neural networks by soft weight-sharing. Neural Computation, vol. 4, 473-493, (1992).

RUMELHART D.E., HINTON G.E., WILLIAMS R.J.: Learning Internal Representations by error propagation. In Parallel Distributed Processing. D.E. Rumelhart, J.L. McClelland eds, MIT Press, vol. 1, 318-362, (1986).
SAS /Stat User's Guide, version 6, 4th edition.
THODBERG H.H.: Ace of Bayes; application of neural networks with pruning. Tech. Rep. n°1132E, Danish Meat Res. Inst., (1993).
WEIGEND A.S.,RUMELHART D.E., HUBERMAN B.A.: Generalization by weight elimination with application to forecasting. In "Neural Information Processing Systems", NIPS'90, R.P. Lippmann, J.E. Moody, D.S. Touretzky eds., Morgan Kaufmann, vol. 3, 875-882, (1991).

Set λ	0	**0.0001**	0.005	0.001	0.1	q	0.85	0.89	**0.92**	0.94
D_m^l	4359	4360	4455	4556	16522		7008	4765	4634	4606
D_m^v	5050	**5029**	5105	5265	19367		7405	5133	**5014**	5035
D_m^t	4672	4669	4743	4933	19979		6962	4766	4698	4729
selected set							7,5	7,5,10	**7,5,10 1**	7,5,10 1,3
" by SAS						7, 10 , 5				
Set λ	**0**	0.001	0.005	0.01	0.10	q	0.92	0.94	**0.945**	0.95
D_m^l	99.67	99.00	97.34	97.34	96.01		99.66	93.02	92.69	99.66
D_m^v	**83.80**	82.70	82.60	81.60	82.70		83.80	84.10	**84.60**	83.80
D_m^t	80.88	80.94	83.32	82.90	82.12		80.88	83.26	83.18	80.88
selected set							17*	19*	**20***	21*
" by SAS						14*				

Figure 1: Performance after `Regularization` (with weight decay): left and `Variable selection` : right (run with λ^* determined in `Regularization`); for time series (top: 10^5 MSE) and waveforms (bottom)

selected set	7,10,5	7,5,10,1	4,5,7,9,10	14*	20*	40
net	3-7-1	4-7-1	5-7-1 10-7-1	14-30-20-3	20-30-20-3	40-30-20-3
D_m^l	4553	4495	4465	4359	94.66	96.66	99.67
D_m^v	4851	4790	4832	5050	76.80	83.60	83.80
D_m^t	4533	4488	4451	4672	79.62	82.82	80.88

Figure 2: performances or networks retrained on selected variables for time series: left (10^5 MSE) and waveforms (right).

Set π	0	**0.4**	0.6	0.7	0.8	q	0.85	0.98	**0.9993**	0.9995
D_m^l	4503	4530	4531	4531	4532		6798	4643	4585	4566
D_m^v	5108	**4857**	4860	4861	4861		7103	4899	**4820**	4838
D_m^t	4768	4575	4578	4578	4579		6759	4673	4556	4566
selected set							7,5	7,5,10	**7,5,10 9,4**	7,5,10 9,4,3

Figure 3: 10^5 MSE after `Regularization` (with gaussian mixture) for time series