

An optimal linear separator for the Sonar Signals Classification task

Juan-Manuel Torres-Moreno^{1,2} and Mirta B. Gordon³

¹ Laboratoire Informatique d'Avignon
Université d'Avignon et des Pays de Vaucluse
BP 1228 84911 Avignon Cedex 09, France

² École Polytechnique de Montréal
CP 6079 Succ. centre-Ville, H3C 3A7 Montréal, Québec Canada

³ TIMC Domaine de La Merci,
38706 La Tronche Cédex France
`juan-manuel.torres@univ-avignon.fr`

Abstract. The problem of classifying sonar signals from rocks and mines first studied by Gorman and Sejnowski has become a benchmark against which many learning algorithms have been tested. We discovered that both the training set and the test set of this benchmark are linearly separable, although with different hyperplanes. Moreover, the complete set of learning and test patterns together, is also linearly separable. We give the weights that separate these sets, which may be used to compare results found by other algorithms.

1 Introduction

It has become a current practice to test the performance of learning algorithms on *realistic* benchmark problems. The underlying difficulty of such tests is that in general these problems are not well characterized, making it thus impossible to decide whether a better solution than the one already found exists.

The Sonar signals classification benchmark, introduced by Gorman et al. [6] is widely used to test machine learning algorithms. In this problem the classifier has to discriminate if a given sonar return was produced by a metal cylinder or by a cylindrically shaped rock in the same environment. The benchmark contains 208 preprocessed sonar spectra, defined by $N = 60$ real values, with their corresponding class. Among these, $P = 104$ patterns are usually used to determine the classifier parameters through a procedure called learning. Then, the classifier is used to class the $G = 104$ remaining patterns and the fraction of misclassified patterns is used to estimate the generalization error produced by the learning algorithm. We applied Monoplane, a neural incremental learning algorithm, to this benchmark. In this algorithm, the hidden units are included one after the other until the number of training errors vanishes. Each hidden unit is a simple binary perceptron, trained with the learning algorithm Minimeror [2].

2 The Incremental Learning Algorithm

2.1 Definitions

Consider a training set of P input-output pairs $\{\xi^\mu, \tau^\mu\}$, where $\mu = 1, 2, \dots, P$. The inputs $\xi^\mu = (1, \xi_1^\mu, \xi_2^\mu, \dots, \xi_N^\mu)$ may be binary or real valued $N + 1$ dimensional vectors. The first component $\xi_0^\mu = 1$, the same for all the patterns, allows to process the bias as a supplementary weight. The outputs are binary, $\tau^\mu = \pm 1$. The Neural Network built by the learning algorithm has a single hidden layer of H binary neurons connected to the $N + 1$ input units, and one output neuron connected to the hidden units. During training, the number of hidden neurons grows until the number of training errors vanishes. After learning, the hidden units $1 \leq h \leq H$ have synaptic weights $\mathbf{w}_h = (w_{h0}, w_{h1} \dots w_{hN})$, w_{h0} being the bias of unit h . The output neuron has weights $\mathbf{W} = (W_0, W_1 \dots W_H)$ where W_0 is the bias.

Given an input pattern ξ that the network has to classify, the state σ_h of hidden neuron h is given by:

$$\sigma_h = \text{sign} \left(\sum_{i=0}^N w_{hi} \xi_i \right); \quad h = 1, \dots, H \quad (1)$$

The network's output ζ is given by:

$$\zeta = \text{sign} \left(\sum_{h=0}^H W_h \sigma_h \right) \quad (2)$$

with $\sigma_0 = 1$ for all the patterns.

2.2 Monoplane algorithm

The Monoplane algorithm [1, 27] constructs a hidden layer in which each appended hidden unit tries to correct the training errors of the previous hidden unit. In fact, the construction of hidden layer is similar to the first layer construction of the parity machine [10, 11]. But, instead of introducing a second hidden layer implementing the parity, our algorithm goes on adding hidden units (if it is necessary). In the case of binary inputs it was proven [10] that a solution exists with at most P hidden neurons. A solution for real valued inputs also exists [5], the upper bound to the number of hidden units being $P - 1$. The proof that a solution with a finite number of units also exists, is found in [5]. Thus, the algorithm Monoplane converges to a finite size network. Clearly, the upper bounds are not tight, and in practice the algorithm constructs very small Neural Networks [27].

The final number H of hidden units depends on the performance of the learning algorithm used to train the individual binary perceptrons. The best solution should endow the perceptron with the lowest generalization error if the training set is LS, and should minimize the number of errors otherwise. Most

incremental strategies use the Pocket algorithm [16]. It has no natural stopping condition, which is left to the user's patience. None of the proposed alternative algorithms as [28] are guaranteed to find the best solution to the problem of learning. The success of our incremental algorithm relies on the use of Minimerror to train the individual units. Minimerror is based on the minimization of a cost function E which depends on the weights \mathbf{w} through the stabilities of the patterns of the training set. If the input vector is ξ^μ and τ^μ the corresponding target, then the *stability* γ^μ of pattern μ is a continuous and derivable function of the weights, given by :

$$\gamma^\mu = \tau^\mu \frac{\mathbf{w} \cdot \xi^\mu}{\|\mathbf{w}\|} \quad (3)$$

where $\|\mathbf{w}\| = \sqrt{\mathbf{w} \cdot \mathbf{w}}$. The stability measures the distance of the pattern to the separating hyperplane normal to \mathbf{w} ; it has a positive sign if the pattern is well classified, negative otherwise. The cost function E is given by:

$$E = \frac{1}{2} \sum_{\mu=1}^P \left[1 - \tanh \frac{\gamma^\mu}{2T} \right] \quad (4)$$

The contribution to E of patterns with large negative stabilities is 1, i.e. they are counted as 1 error, whereas the contribution of patterns with large positive stabilities is vanishingly small. Patterns within a window of width $\approx 2T$ centered on the hyperplane contribute to the cost function even if they have positive stability, proportionally to $1 - \gamma/T$. It may be shown that E may be interpreted as a noisy measure, at temperature T , of the number of training errors [3]. The properties of its global minimum, studied theoretically with methods of statistical mechanics [4], have been confirmed by numerical simulations [2, 7]. In particular, the minimum of E in the limit $T \rightarrow 0$ corresponds to the weights that minimize the number of training errors. If the training set is LS, the weights that separate the training set are not unique. It was shown that there is an optimal learning temperature such that the minimum of the cost function endows the perceptron with a generalization error numerically indistinguishable from the optimal (bayesian) value.

The algorithm Minimerror minimizes the cost E through a gradient descent, combined with a slow decrease of the temperature T equivalent to a deterministic annealing [2, 7], and determines automatically the optimal temperature at which it has to stop.

3 The Sonar Benchmark

The set of examples contains 111 patterns obtained by bouncing sonar signals off a metal cylinder at various angles, and 97 patterns obtained from rocks under similar conditions. Each pattern is a set of 60 numbers in the range [0,1]. Each number represents the energy within a particular frequency band, integrated over a certain period of time. The label associated with each pattern contains

the number $\tau = +1$ if the signal correspond to a rock and $\tau = -1$ if it is a mine (metal cylinder).

SET	N	P	G	$\tau = +1$	$\tau = -1$
<i>Train</i>	60	104	104	55	49
<i>Test</i>	60	104	104	42	62
<i>Sonar</i> (<i>Test</i> + <i>Train</i>)	60	208	0	97	111

Table 1. Number of patterns and distribution of classes

We have numbered each patterns with a label absolute μ . Of this way, the set *Train* has μ of 1 to 104 and the set *Test*, μ of 105 to 208. This identification allows to analyze each pattern of way individual. The used procedure was the following: learning the P patterns of *Train* set and measure the error of generalization on the G patterns of *Test* set. Later, learn on the P patterns of *Test* set and measure generalization over G patterns of *Train* set. Finally take the *Sonar* set (*Train* + *Test*) and try to learn it. In this last case, of course there is not possibility of measuring generalization. We have carried out a pre-processing of the vector ξ of the learning set, through the following normalization:

$$\xi_i^\mu \leftarrow \frac{\xi_i^\mu - \langle \xi_i \rangle}{\sigma_i} \quad (5)$$

$$\langle \xi_i \rangle = \frac{\sum_{\mu=1}^P \xi_i^\mu}{P} \quad (6)$$

$$\sigma_i = \frac{\sum_{\mu=1}^P (\xi_i^\mu - \langle \xi_i \rangle)^2}{P} \quad (7)$$

Some authors [24, 25] have reported that the learning set *Train* is linearly separable. But most of people, report results obtained through the backpropagation algorithm (or their variants), and they find too complex nets: with a number excessive of parameters [26](weights and units).

We found [1] that we needed two hidden units to learn without errors the training set *Train*, but the generalization error was lower with only one unit (a simple perceptron) than with two hidden units. This result is usually considered as overfitting, a not well defined category used to describe this kind of behaviour.

A finer tuning of the parameters of Minimererror showed however that this benchmark is linearly separable. In fact, the *Train* set, the *Test* set, and *both* sets together (i.e. the $P + G = 208$ in *Sonar* set) patterns are linearly separable. In Tables 2, 3 and 4, we give the values for the weights of the perceptron that separate each of the three sets *Train*, *Test* and *Sonar*.

We have calculated in table 5 the cosine of the angle α between the vector W_{Sonar} that separates the whole set, and the W_{Train} , W_{Test} vectors, that separate *Test* and *Train* set respectively. Also the cosine between W_{Train} and W_{Test} is calculated, following equation (8).

$$W_{Train} = \{$$

$$\begin{aligned} & -0.0692, -1.5031, -1.9481, -0.2835, -1.0162, -0.2870, -0.5139, -0.3040, \\ & 2.3106, -0.4349, -0.6610, -1.0995, -1.2447, -1.3281, -0.7392, 0.6469, \\ & 1.7862, 1.2227, -0.0513, -0.6431, -0.8745, -0.8290, -0.8084, -0.6578, \\ & -1.0453, -1.2332, -0.9860, -1.0617, -1.0097, -1.3597, -0.5245, 1.6822, \\ & 0.6588, -0.1056, -0.0794, 0.2998, 1.2290, 0.6709, -0.3025, 0.2681, \\ & 1.2375, 0.2485, 0.1098, 0.1693, -0.5717, -1.2458, -0.7116, -0.1323, \\ & -1.3481, -2.6467, 1.0464, -0.7163, -0.8324, -0.4364, -1.1849, 1.3439 \\ & 0.4299, 1.0813, -0.9662, -0.3129, 0.0015 \\ & \} \end{aligned}$$

Table 2. Weights of Minimerror trained perceptron for *Train* test

$$W_{Test} = \{$$

$$\begin{aligned} & -0.4035, -0.9738, 1.0107, 0.9301, -0.8997, -0.5649, 0.9318, 1.5102, \\ & 0.0477, -1.6914, -1.3137, -2.0763, -2.0756, -0.5307, 0.8317, 1.4271, \\ & 0.6112, 0.5119, 0.2081, -0.8285, -1.4488, -1.4337, -1.1908, -1.0213, \\ & -0.3653, 0.2701, 0.2465, -0.2028, -0.3975, -0.2049, 0.1843, 1.2486, \\ & 0.3270, 0.2806, 0.4427, 0.7089, 1.5015, 1.5818, 0.2483, -0.6511, \\ & 0.6822, 0.4056, -0.4476, -1.4451, -2.1873, -1.5600, -1.0694, -0.6042, \\ & -0.5170, -0.1298, 1.0330, -1.3454, -1.6560, 0.1098, -0.1249, -0.0331, \\ & -0.1748, 0.2088, -0.7949, -1.7304, 0.1419 \\ & \} \end{aligned}$$

Table 3. Weights of Minimerror trained perceptron *Test* set

$$W_{Sonar} = \{$$

$$\begin{aligned} & -0.0290, -0.7499, -0.0626, 0.5991, -0.1493, 0.2057, -0.3432, 0.5235, \\ & 0.2407, -0.2480, -0.2069, 0.4854, -2.0100, 0.7256, 0.0868, -0.6282, \\ & 1.0116, 0.8600, -0.8842, -0.1056, -1.4125, 1.7174, -2.1101, 0.3378, \\ & -0.8198, -0.1804, 1.4065, -2.2840, 1.4039, -0.1153, -2.6714, 3.3527, \\ & -1.0352, -1.1619, 1.4134, -0.6482, 0.3479, 0.9895, -0.3477, -0.5707, \\ & 1.2758, -0.6628, 0.6288, -0.7920, -0.0850, -0.1348, -0.7794, 0.2451, \\ & -0.8392, -0.5660, 1.4128, -0.4471, -0.5439, -0.2079, -0.1840, -0.0060, \\ & 0.2276, -0.0158, -0.2637, -0.1579, 0.1238 \\ & \} \end{aligned}$$

Table 4. Weights of Minimerror trained perceptron that separates the Full *Sonar* (*Train* + *Test*) dataset

$$\cos(\alpha) = \frac{\mathbf{W}_a \cdot \mathbf{W}_b}{(N+1)^2} \quad (8)$$

	(W_{Sonar}, W_{Train})	(W_{Sonar}, W_{Test})	(W_{Train}, W_{Test})
$\cos(\alpha)$	0.51615	0.34238	0.4

Table 5. Cosine

In Table 6 at right, we give the distances of each pattern bad classified by W_{Train} to the hyperplane separator W_{Sonar} . $\epsilon_g = 19.2$ (15 F+ 5 F-). At left, we give the distances of each pattern bad classified by W_{Test} to the hyperplane separator W_{Sonar} . $\epsilon_g = 23.1$ (5 F+ 19 F-)

<i>Test set</i>				
i	μ	Field	$\gamma(W_{Sonar})$	τ^μ
1	105	1.19697e-01	2.09029e-03	-1
2	107	7.97467e-02	1.87343e-03	-1
3	108	1.19431e-01	1.43453e-02	-1
4	109	3.59889e-02	2.09760e-03	-1
5	110	1.95963e-02	6.21865e-04	-1
6	111	6.05680e-02	3.18180e-04	-1
7	118	2.02768e-02	8.74281e-03	-1
8	122	3.07859e-02	2.66651e-02	-1
9	131	6.87472e-02	7.54780e-03	-1
10	133	1.37587e-02	5.23483e-03	-1
11	135	4.35705e-03	3.23781e-04	-1
12	136	7.91603e-03	1.01329e-02	-1
13	138	2.35263e-02	1.13658e-02	-1
14	142	2.22331e-02	7.53167e-03	-1
15	143	2.36318e-02	6.63512e-03	-1
16	168	-1.34434e-02	8.57956e-03	1
17	170	-8.20828e-02	1.96977e-03	1
18	197	-4.87395e-02	7.08260e-05	1
19	202	-2.91468e-03	1.02479e-02	1
20	203	-8.11795e-02	4.08223e-02	1

<i>Train set</i>				
i	μ	Field	$\gamma(W_{Sonar})$	τ^μ
1	5	1.60234e-02	1.99901e-03	-1
2	6	2.76646e-02	1.98919e-03	-1
3	9	1.63374e-02	5.77784e-05	-1
4	26	1.90089e-02	3.73223e-05	-1
5	39	5.77398e-02	2.28692e-04	-1
6	51	-5.05614e-02	3.23683e-02	1
7	53	-1.35299e-01	2.60559e-03	1
8	55	-4.13985e-02	2.24705e-03	1
9	57	-7.71201e-02	2.30675e-03	1
10	58	-2.70548e-02	2.46977e-03	1
11	61	-3.02880e-02	2.57994e-03	1
12	62	-3.16819e-02	1.94132e-02	1
13	64	-6.23916e-02	7.15758e-03	1
14	65	-4.00952e-02	2.49840e-03	1
15	66	-2.10826e-01	2.43180e-03	1
16	72	-7.44215e-02	2.31141e-02	1
17	73	-2.71180e-02	2.67136e-02	1
18	77	-1.34531e-01	2.44142e-03	1
19	82	-2.26242e-01	1.82462e-03	1
20	83	-5.91598e-02	2.07447e-03	1
21	84	-5.80561e-02	4.04605e-04	1
22	97	-4.57645e-02	4.56010e-04	1
23	98	-6.72313e-02	1.06360e-04	1
24	100	-1.83484e-02	3.35972e-03	1

Table 6. Bad patterns over *Test* and *Train* sets

4 Discussion and Conclusion

In this paper, we have shown that the sonar benchmark is linearly separable. Both sets, Train and Test are it but for hyperplanes different, it generates a certain ϵ_g . We have found solutions to the three sets using the perceptron learning rule [29], and we have found that the generalization error is superior than the error met with Minimerror. The weight vector for the complete set, *Sonar*, could be used like test for learning algorithms able to find the best separator hyperplane.

References

1. Torres-Moreno J. M., Peretto P. and Gordon M. B. 1995. An evolutive architecture coupled with optimal perceptron learning for classification. In ESANN'95, Brussels, M. Verleysen ed., D-facto, pp 365-370.
2. Gordon M. B. and Berchier D. 1993a. Minimerror : A perceptron learning rule that finds the optimal weights, in ESANN'93. Brussels, D facto, pp.105-110.
3. Gordon M. B., Peretto P. and Berchier D. 1993b. Learning algorithms for perceptrons from statistical physics. J.Phys.I France 3, 377-387.
4. Gordon M. B. and Gempel D. 1995. Optimal learning with a temperature dependent algorithm. Europhys.Lett. 29(3), 257-262.
5. Gordon M. B. 1996. Convergence learning. J.Phys.I France 3, ICNN' 96377-387.
6. Gorman R. P. and Sejnowski T. 1988. Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets. Neural Networks 1, 75-89.
7. Raffin B. and Gordon M. 1995. Learning and generalization with Minimerror, a temperature dependent learning algorithm. Neural Comp. 7(6), 1182-1200.
8. Mézard M. and Nadal J.-P. 1989. Learning in feedforward layered neural networks: the tiling algorithm. J.Phys.A 22, 2191-2203.
9. Frean M. 1990. The upstart algorithm: a method for constructing and training feedforward neural networks. Neural Comp. 2, 198-209.
10. Martinez D. and Estève D. 1992. The offset algorithm: building and learning method for multilayer neural networks. Europhys. Lett. 18, 95-100.
11. Biehl M. and Oppen M. 1991. Tilinglike learning in the parity machine. Phys. Rev. A 44, 6888.
12. Marchand M., Golea M. and Ruján P. 1990. A convergence theorem for sequential learning in two-layer perceptrons. Europhys. Lett. 11, 487-492.
13. Fahlman S. E. and Lebiere C. 1990. The cascade-correlation learning architecture., in Advances in Neural Information Processing Systems 2. San Mateo, CA., Morgan Kaufmann, pp.574-532.
14. Sirat J. A. and Nadal J. P. 1990. Neural trees: a new tool for classification. Network 1, 423-438.
15. Alpaydin E. A. I. 1990. Neural models of supervised and unsupervised learning. Ph. D. Thesis EPFL 863, Lausanne, Suisse.
16. Gallant S. I. 1986. Optimal linear discriminants. IEEE Proc. 8th. Conf. Pattern Recognition, Paris, Oct. 28-31, pp. 849-852.
17. Vapnik V. 1992. Principles of risk minimization for learning theory. In Neural Information Processing Systems, Touretzky D. S. ed., Morgan Kaufmann, San Mateo, CA., vol.4, pp.831-839.

18. Geman S., Bienenstock E. and Doursat R. 1992. Neural networks and the bias/variance dilemma. *Neural Computation* 4, 1–58.
19. Prechelt L. 1994. Proben-1, a set of neural network benchmarks problems and benchmarking rules. Technical Report 21/94. Fakultät für Informatik, Universität Karlsruhe, Germany.
20. Solla S.A. 1989. Neural networks from models to applications. Ed. L. Personnaz and G. Dreyfus, I.D.S.E.T., Paris 1989, p. 168-177
21. Knerr S., Personnaz L. and Dreyfus G. 1990. *Neurocomputing: Algorithms, Architectures and Applications*. F. Fogelman ed. Springer-Verlag.
22. Nadal J.P. 1989. Study of a growth algorithm for a Feedforward Network. *J. Phys. A* 22, 2191-2203.
23. Merz, C.J., and Murphy, P.M. (1996). UCI Repository of machine learning databases [<http://www.ics.uci.edu/mlearn/MLRepository.html>]. Irvine, CA: University of California, Department of Information and Computer Science.
24. Fahlman, S., Hoehfeld, M., (1991), Learning with Limited Numerical Precision Using the Cascade-Correlation Algorithm. Report CMU-CS-91-130, Carnegie Mellon University.
25. Roy, A., Kim, L.S. and Mulkhopadhyay, S. (1993), A polynomial time algorithm for the construction and training of a class of multilayer perceptrons. *Neural Networks*, Vol. 6, No. 4, pp. 535-545.
26. Verma, B.K., Mulawka, J.J., (1995) A new Training Algorithm for Feedforward Neural Networks, In: *ESANN'95*, M. Verleysen ed. pp.359-364.
27. Torres-Moreno, J.M. (1997) *Apprentissage et généralisation par des réseaux de neurones : étude des nouveaux algorithmes constructifs*. PhD Thesis, INPG, Grenoble, France.
28. Frean, M., (1992) A “thermal” perceptron learning rule, *Neural Computation* 4:6, 946–957.
29. Hertz, J., Krogh, A., and Palmer, R.G., (1991) *Introduction to the theory of Neural Computation*. Addison-Wesley Publishing Co.