SELECTING VECTORS FROM A TRAINING SET FOR BACKPROPAGATION NETWORKS

Angel Gutierrez
Department Mathematics and Computer Science
Montclair State College
Upper Montclair, NJ 07043, USA

ABSTRACT

Backpropagation neural networks could be used to learn a mapping f from R^n to R^m. A set of examples, the training set, is available to achieve this goal. The learning process is carried out by presenting, to the network, elements from the training set, and modifying the weights according to the error on the output. These elements are selected using a fixed probability density function. This paper presents a new method, assuming that there is a uniform probability density function, but taking into account the varying difficulty of learning for each component. This new method produces smaller errors on almost all the cases.

Keywords : Neural networks, backpropagation networks, vector-valued functions, selection on training sets.

INTRODUCTION

A neural network with a regular configuration, according to [7], of two input nodes, two output nodes and a hidden layer containing s nodes, - see Figure 1 - is used to learn the mapping $f: A \subsetneq R^2 \dashrightarrow R^2$, $y = f(x)$, that represents the conversion from Cartesian to polar coordinates. If the neural network output is represented by $Y(x,w)$, where w is the set of weights associated with the nodes of the network, then a measure of the error between the actual values $f(x)$ and the one supplied by the network is given, according to [3], by

$$F(w) = \lim_{N\to\infty}\left(\frac{1}{N}\right)\left(\sum_{k=1}^{k=N} \|f(x_k) - Y(x_k, w)\|^2\right) \qquad (1)$$

The learning process, see reference [4], is accomplished by changing the value of the weights during the training period, according to the formula

$$w_{ji}(k+1) = w_{ji}(k) - \eta_k \times \frac{\partial F}{\partial w_{ji}} \qquad (2)$$

that is the known jump every time variant of the generalized delta learning rule [5],[10]. The generic weight w_{ji} of the network represents the weight associated with the information leaving the node i and entering the node j in the forward mode, and η_k is the learning rate at the step k. The output of the hidden layer is filtered through the sigmoidal function

$$y = \frac{1.0}{1.0 + e^{-x}}$$

to control the nonlinear terms, and the output layer provides the approximation to the two-dimensional mapping under consideration. We add a bias term to both input and hidden layers to make the neural network more powerful.

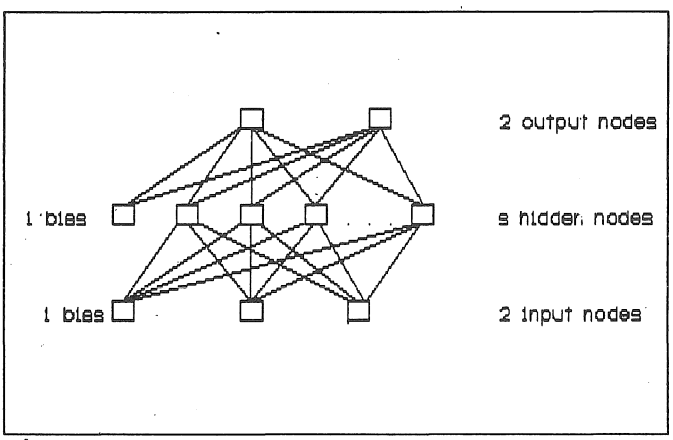

Figure 1 : Neural network

The vector-valued function is seen by a network as a single entity, and the vectors in the training set have the same probability to be chosen. It is reasonable to think that if we choose the vectors taking into account the varying difficulty of learning for each component, a better

result could be obtained. Therefore, when choosing vectors form the training set, we use a technique that resembles an inverse simulated annealing [1]. Each training vector is assigned a probability of acceptance based on its difficulty to be learned. Then each vector is rejected with a low uniform probability, unless its acceptance function removes the rejection.

This idea is studied in this paper varying the number s of nodes in the hidden layer in one case and changing the seed for the random number in another. In the next two sections we set up the working conditions for our examples and exhibit the obtained results. In the last section the conclusions are drawn.

SENSITIVITY ANALYSIS

The square $A \equiv \{ (x,y) \mid -1 \leq x \leq 1 , -1 \leq y \leq 1 \}$ is used as the original domain for the conversion mapping to be learned. So given $(x,y) \in A$, the map under consideration is

$$radius=\sqrt{x^2+y^2}$$

and

$$\theta = \begin{cases} \arctan\frac{y}{x} & for\ x>0, y>0 \\ \arctan\frac{y}{x}+2\pi & for\ x>0, y<0 \\ \arctan\frac{y}{x}+\pi & for\ x<0 \end{cases}$$

with obvious values for the 4 semiaxes.

Each network, containing s regular nodes and one bias node in the hidden layer is trained with a fixed set of T points (vectors) in the unit square. The networks were first trained using the correct output for each input vector, using a uniform probability density function. Then, in order to replicate the experimental condition the same networks where trained with the new method. When each learning process was completed, i.e. the number of predetermined presentations were concluded, the error in learning the training set was computed, using the

formula 3,

$$e = \frac{1}{T}\left(\sum_{k=1}^{k=T} \| f(x_k) - Y(x_k, w) \|^2\right) \qquad (3)$$

which is a modification of the mean square error function, where x_k are the training points. Since overtraining is a normal phenomenon in backpropagation neural networks, the trained networks were tested with a set of P new points. For this new set we compute the new error, using a formula equivalent to (3), with T replaced by P, i.e.

$$e_0 = \frac{1}{P}\left(\sum_{k=1}^{k=P} \| f(x_k) - Y(x_k, w) \|^2\right) \qquad (4)$$

At the same time, since the new method is supposed to have different learning ability for each output component, two new errors are computed, according to the formula

$$e_j = \frac{1}{P}\left(\sum_{k=1}^{k=P} | f_j(x_k) - Y_j(x_k, w) |^2\right) \qquad j=1,2 \qquad (5)$$

where e_1 represents the error on the radius, and e_2 is the error on the angle.

Each training vector was selected if the random uniform variate obtained for it was below q, a fixed value during the training, with $0 < q \leq 1$. If the vector was rejected by the previous criterion, it could be accepted if a new variate was below f(p), with $0 < f(p) \leq 1$. Here f(p) is a given function, which depends upon p, a variable that in some way denotes the learning difficulty associated to that vector.

EMPIRICAL RESULTS

The experiments were conducted using a set of 500 training points. The learning rate η was given by the linear function

$$\eta_k = \eta_0 + (\eta_1 - \eta_0) \times \frac{k}{100000}$$

where k is the iteration step, η_0 is the initial learning rate, set at 0.050, and η_1 is the final rate, set at 0.001. These values for the initial and final learning rate were known to be

acceptable [2]. The guided selection of the training set was established by the following function

$$r<q \vee s<f(p)$$

where r and s are random uniform variates and

$$f(p)=1+p\times(-1.2+0.6\times p)$$

p being defined as

$$p=\frac{a}{\pi}$$

and a is the angle associated to the training vector. In other words, we are assuming that the angle is more difficult to be learned when it is near 0 and 2π radians. The function f(p) tries to represent quantitatively this difference, but it was chosen arbitrarily. If the new method worked with this unsophisticated function, it gives a guarantee that it will work if a more elaborated function is used.

The testing set contained 18018 points chosen in the following way : The radius was changed from 0.1 to 1 using increments of 0.01; for each value of the radius the angle will take two set of values. The first set contains the angles from $\pi/51$ to $99\pi/51$ using increments of $\pi/51$ radians. The second one contains the angles from $\pi/50$ to $99\pi/50$ using increments of $\pi/50$ radians. Using a large set of testing vectors will avoid the appearance of sampling errors in our computation of the quadratic and component errors.

Since a network with more hidden nodes does not necessarily perform better [6],[10], the experiment was started using 24 hidden nodes, and then the number of nodes was diminished, until a configuration that contained 12 hidden nodes was reached, see reference [2].

In this first experiment the hidden nodes were changed, but the seed for the random number generator was 123451, the same for all of them. The number of presentations of the training vectors also remained constant with a value of 100000. The weights were initialized using

random values in the interval [0,1). The results of this experiment are shown on Table 1, although the error e_0 in this and all the other tables is shown with a magnification of two, to see the differences more clearly. That means that the actual error will be obtained from the table by dividing by two.

s	e	e_0	e_1	e_2	q	e	e_0	e_1	e_2
24	0.156	0.265	0.046	0.276	0.8	0.153	0.257	0.046	0.276
20	0.153	0.259	0.036	0.284	0.8	0.154	0.259	0.035	0.281
18	0.141	0.247	0.034	0.264	0.8	0.141	0.231	0.042	0.261
16	0.176	0.310	0.066	0.286	0.8	0.179	0.310	0.068	0.286
14	0.189	0.351	0.084	0.303	0.8	0.190	0.348	0.083	0.303
12	0.215	0.313	0.075	0.272	0.8	0.167	0.282	0.064	0.275

Table 1

In the second set of experiments, the number of hidden nodes was 14, but now the random seed was changed. The number 14 was chosen because it represents, according to [8], a good value for our number of training points. The number of presentations of training vectors was increased to 200000 and the weights were initialized using random values in the intervals [0,1) and (-1,0]. Finally the value of q was changed to 0.9 in all the guided learning cases. In this case the results can be found on Table 2, 3 and 4, where the values on the left are obtained by the regular training method, and the values on the right are obtained with the new method.

e	e_0	e_1	e_2	Seed	e	e_0	e_1	e_2
0.160	0.250	0.031	0.274	772241	0.156	0.241	0.036	0.266
0.128	0.202	0.044	0.216	335511	0.129	0.199	0.042	0.216
0.148	0.252	0.064	0.234	554431	0.147	0.241	0.059	0.233
0.112	0.182	0.033	0.208	643231	0.113	0.181	0.033	0.203
0.124	0.206	0.036	0.231	16777216	0.123	0.202	0.036	0.231
0.146	0.229	0.053	0.233	223341	0.145	0.225	0.052	0.232

Table 2

e	e_0	e_1	e_2	Seed	e	e_0	e_1	e_2
0.153	0.250	0.051	0.250	3445423	0.157	0.250	0.051	0.252
0.130	0.213	0.053	0.219	225543	0.130	0.206	0.052	0.212
0.118	0.186	0.035	0.216	224453	0.118	0.183	0.031	0.215
0.127	0.215	0.026	0.246	2244131	0.128	0.211	0.025	0.238
0.130	0.222	0.055	0.228	225533	0.131	0.217	0.055	0.228
0.117	0.200	0.032	0.222	2125533	0.115	0.197	0.033	0.222

Table 3

e	e_0	e_1	e_2	Seed	e	e_0	e_1	e_2
0.135	0.226	0.065	0.213	131	0.135	0.224	0.066	0.209
0.150	0.245	0.063	0.233	13457	0.149	0.241	0.063	0.231
0.150	0.223	0.028	0.247	11334457	0.152	0.232	0.028	0.255
0.125	0.205	0.036	0.223	331155	0.120	0.191	0.032	0.215
0.165	0.265	0.053	0.255	334551	0.162	0.257	0.049	0.263
0.102	0.175	0.027	0.207	345523	0.102	0.175	0.026	0.206

Table 4

CONCLUSIONS

A new method to select vectors from a training set has been compared with the regular one. The results on all tables corroborate the conjecture that the new method performs equal to or better than the regular one. The only exception occurs in table 4, for the seed 11334457. The value of these errors lies in the middle range of all found errors, 0.175:0.265. When the value of q was diminished to 0.8 for this particular case, the error of the new method diminished to 0.230, but still remained greater than the one obtained using the regular method.

In the cases where the errors e_0 were the same, like in table 4, case of seed 345523, using a new value of q = 0.8, the error e_0 was diminished to 0.170, with e_1 = 0.024 and e_2= 0.204, an improvement over the previous results. Therefore it can be said that the lowest errors of all the configurations correspond to a network trained with the new method using a q value of 0.8.

The results of the experiments are promising, but more detailed comparisons are to be made to determine the influence of the function f(p) chosen, of the value of q, and of the number of presentations, on the behavior of the networks with respect to each method.

BIBLIOGRAPHY

1. I.O. Bohachevsky, M.E. Johnson and M.L. Stein,"Generalized Simulated Annealing for Function Optimization", Technometrics, 28, 209-217, 1986.

2. A. Gutierrez, "Learning Simulation of Vector-Valued Functions", Proc. of the Twenty-Second Annual Pittsburgh Conference on Modeling an Simulation, 22, 1938-1943, 1991.

3. R. Hecht-Nielsen, "Neurocomputing", Addison-Wesley, 1989.

4. R. Hecht-Nielsen, "Theory of the Backpropagation Neural Networks", Proc. of the International Joint Conference on Neural Networks, I, 593-611, 1989.

5. M. Hirsch, "Dynamical Systems Review", tutorial presented at the 1988 IEEE International Conference on Neural Networks.

6. E.A. Rietman, "Experiments in Artificial Neural Networks", Tab Books Inc., Blue Ridge Summit, PA, 1988.

7. D.E. Rumelhart, G.E. Hinton and R.J. Williams,"Learning Internal Representations by Error Propagation", in Parallel Distributed Processing: Explorations in The Microstructure of Cognition, Cambridge, MA, MIT Press, 1986.

8. SARA: Entorno para el Desarrollo de Aplicaciones de Redes Neuronales, Manual del Usuario, ADIC-IIC, Madrid, 1992.

9. F.M. Silva and L.B. Almeida,"Speeding up backpropagation", in Neural Networks for Sensory and Motor Systems, R. Eckmiller (ed.), North-Holland, 1990.

10. H. White, "Learning in Artificial Neural Networks: A Statistical Perspective", Neural Computation, 1, 425-464, 1989.