CONTEXT DEPENDENT LEARNING IN NEURAL NETWORKS

L.J. Spreeuwers, B.J. van der Zwaag, F. van der Heijden

University of Twente*, The Netherlands

ABSTRACT

In this paper an extension to the standard error backpropagation learning rule for multi-layer feed forward neural networks is proposed, that enables them to be trained for context dependent information. The context dependent learning is realised by using a different error function (called AVerage Risk: AVR) instead of the sum of squared errors (SQE) normally used in error backpropagation and by adapting the update rules. It is shown that for applications where this context dependent information is important, a major improvement in performance is obtained.

Key words: neural networks, learning, context dependency

INTRODUCTION

The error backpropagation learning rule, as described by Rumelhart, Hinton and Williams [13] uses the squared error (SQE) as an error measure. The error backpropagation learning rule is an optimisation method that is used to minimise this error measure. In many cases the application of the least squares criterion results in optimal or acceptable suboptimal solutions. The accuracy of the approximation of the resulting behaviour to the desired behaviour of a neural network depends on the following factors:

- quality of the used training set
- size and architecture of the used network
- optimisation method
- optimisation criterion

Papers addressing the influence of training sets are among others: [8, 12, 18]. A paper addressing the problem of choosing the optimal size for the network for a certain application is [9] and some enhancements to error back-propagation and other optimisation methods for feed forward neural networks can be found in [1, 7, 11].
In [6] an alternative error measure is proposed based on a logarithmic combination of the actual and target outputs. The authors show that this improves generalisation, reliability and convergence speed of the training process. In [10] an error measure is proposed that minimises the probability on classification errors, rather than the SQE. Neither of these error measures takes into account different types of errors and the costs of the errors that are typical for specific applications. This research was initiated because we wanted to design neural network edge detectors [14, 15] that were to ignore certain types of errors and avoid others in accordance with the application. The performance measure we use for the evaluation of edge detectors, the average risk (AVR) evaluation method [4, 16, 17], takes into account different types of errors and weights them in accordance with the application. E.g. if the objective of an image processing system is to recognise an object in an image, it may be acceptable if the edges are not detected on their exact position as long as they are detected reliably. Thus, in this case small displacement errors can be safely ignored. The squared error measure that is used in the error backpropagation learning rule, however, appears to penalise these displacement errors very heavily, as will be explained in detail later.
The next sections are addressed to the analysis of when the least squares error criterion does not result in a correct solution and how it can be replaced by a more appropriate optimisation criterion. A neural network for pulse detection in a one dimensional signal is used as an illustration of choosing an error measure matched to the application. As a more extensive test, neural network step detectors were compared to a number of other step detectors.

NON-OPTIMAL LEAST SQUARES SOLUTIONS

If the training set is of good quality and there are no generalisation problems, optimisation for the SQE can still yield neural networks that do not exhibit the desired behaviour. E.g. a neural network can exhibit the desired behaviour while the SQE is relatively high. It is even possible that a network approximates the desired behaviour better for certain higher values of the SQE error measure. This is e.g. the case if the SQE error measure punishes heavily certain types of errors that are in fact less serious. The reverse is also possible, that the SQE is not sensitive for or too insensitive for certain types of errors. The problem and solution presented in this paper refer to these types of defects of the SQE as an error measure. The above described problem is very likely to turn up in cases where the input patterns of a neural network are not independent, but have a mutual relation. This is the case e.g. in signal and image filtering with neural networks where a small neural network "scans" the signal or image. As an example, consider a neural network with 3 inputs that has to detect pulses in a one dimensional digital signal (see

*Department of Electrical Engineering, Control, Systems and Computer Engineering Group, Laboratory for Measurement Science and Instrumentation, P.O. Box 217, 7500 AE Enschede, The Netherlands, E-mail: luuk@mi.el.utwente.nl

Image Processing And Its Applications, 4–6 July 1995
Conference Publication No. 410, © IEE 1995.

fig.1).

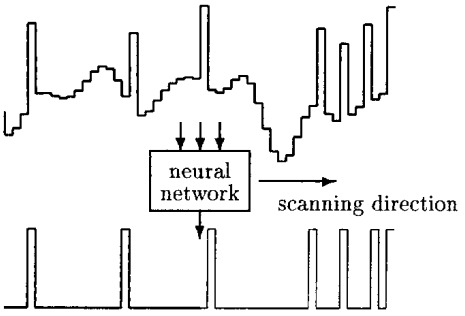

FIGURE 1: Pulse detection with a neural network

In order to detect if there is a pulse at time t in the signal, the sample at time t and the neighbouring samples are presented as input pattern to the network. The single output of the network should be 1 if there is a pulse at time t and 0 if not. In this case the relation between the input patterns is a temporal relation. The input patterns presented to the neural network pulse detector cannot simply be considered as independent patterns, but have a mutual relation, which also determines the types of errors that can occur. Suppose that training data is available of which the pulses have been localised (e.g. by human inspection) with an accuracy of 1 sample period. This means that if for a sample in the training set the target is 1 (a pulse is present), it might have been the previous or the next sample that actually is the position of the pulse. Now consider the effect of this on the squared error. In fig.2 the target output t of sample p is 1, but actually the pulse is at position $p-1$. The network correctly detected the pulse at position $p-1$, i.e. its output y for input pattern $p-1$ is 1 and for p it is 0. As shown in fig.2 this results in a contribution to the total squared error of 2 for these two input patterns.

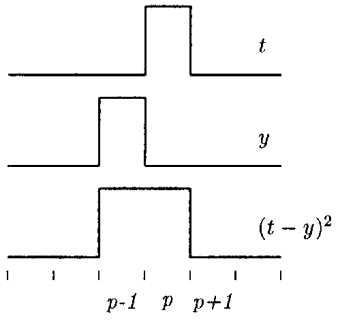

FIGURE 2: Double counting of displacement errors

Thus the error due to a one sample period displacement of the pulse is counted twice. Since the SQE due to simply missing the pulse is only 1, the network will tend to suppress the pulse. Therefore, a network that is trained for minimum SQE for this problem will probably fail to detect many of the pulses.

The conclusion that can be drawn from this is that an error measure for a certain application should be able to differentiate between the different types of errors that can occur for this application and can take into account the seriousness of the different errors (generally called *costs*). The application of the SQE error measure in this light is rather limited.

USE OF OTHER OPTIMISATION CRITERIA IN ERROR BACK PROPAGATION

The derivation of the error backpropagation learning rule by Rumelhart et al. [13], begins with the introduction of the error measure:

$$E = \frac{1}{2}\sum_p \sum_i (t^{pi} - y^{pi})^2 \qquad (1)$$

with: t^{pi} the target output for input pattern x^p, output i, y^{pi} the actual output and the summations are over all outputs i and training samples p.

For a multi-layer feed forward network Rumelhart et al. [13] derived the following weight update rule for a gradient descent optimisation method:

$$\Delta w_k^{pij} = \eta \delta_k^{pi} o_{k-1}^{pi} \qquad (2)$$

with Δw_k^{pij} the weight update of a connection between processing element j in layer $k-1$ to processing element i in layer k, η the learning rate, δ_k^{pi} an error measure local to the processing element i in layer k and o_{k-1}^{pj} the output value of processing element j in layer $k-1$ for input pattern x^p.

The δ's in eq. 2 are determined recursively. For the output layer (layer L) the δ's can be calculated directly from the error $t^{pi} - y^{pi}$:

$$\delta_L^{pi} = (t^{pi} - y^{pi}) f'(net_L^{pi}) \qquad (3)$$

For each lower layer, the δ's can be calculated from the δ's in the higher layers:

$$\delta_k^{pi} = \sum_h \delta_{k+1}^{pi} w_{k+1}^{hi} f'(net_k^{pi}) \qquad (4)$$

with $f'()$ the derivative of the transfer function of a processing element, net_k^{pi} the weighted sum of the inputs of processing element i in layer k for input pattern x^p and the summation is over all processing elements h in layer $k+1$.

The question now is how do these equations change if another error measure than eq. 1 is used. Since the δ's in the lower layers are calculated from the δ's in the higher layers, it follows that the error measure is only explicitly present in the calculation of the δ's of the output layer. If eq. 1 is not substituted into the calculation of the δ's of the output layer, in stead of eq. 3 the following expression is obtained:

$$\delta_L^{pi} = -\frac{\partial E}{\partial o_L^{pi}} f'(net_L^{pi}) \qquad (5)$$

Note that since L is the last (output) layer, $y^{pi} = o_L^{pi}$.

This is a general formula for the calculation of the δ's of the output layer, for any error measure E. Thus other error measures require only a slight modification of the learning rule.

AVERAGE RISK, AN ERROR MEASURE MATCHED TO THE APPLICATION

The average risk (AVR) [3, 16, 17] is an error measure that takes into account the different types of errors, their costs and their probabilities. The average risk for a certain type of error is defined as the product of the probability of this type of error and its cost. The average risk of all errors is the sum of the average risks of the individual errors:

$$AVR = \sum_i P(\epsilon_i)\lambda(\epsilon_i) \tag{6}$$

with $P(\epsilon_i)$ the probability of error type ϵ_i, $\lambda(\epsilon_i)$ its cost, and the summation is over all error types i.

The probabilities of the different error types can be estimated from a training set. Thus, for a given set of costs: $\vec{\lambda}$, the AVR can be estimated from the training set. A neural network should be optimised for the average risk in order to obtain the best approximation of the desired behaviour.
From eq.5 it is clear that the error measure for error backpropagation must be a continuous and differentiable function in o_L^{pi}, the output of the network. Because the average risk error measure of eq. 6 is neither, in order to train a neural network for minimum average risk, an approximation of eq. 6 must be derived that is continuous and differentiable. Below an approximation is derived that makes use of the assumption that for classification problems at the end of the learning process for most input patterns the outputs of the neural network will be either close to 0 or close to 1. The method is best explained by first analysing the SQE error measure under these assumptions.
Consider a 2 class problem that is to be solved with a neural network with a single output, that should be 0 for class 0 and 1 for class 1. The squared error of this network is:

$$SQE = \sum_p (t^p - y^p)^2 \tag{7}$$

Because the desired output t^p can only be 0 or 1, the right term can be split into two parts:

$$SQE = \sum_{p|t=0} (y^p)^2 + \sum_{p|t=1} (1 - y^p)^2 \tag{8}$$

If it is assumed that y^p is always close to 0 or 1, the following approximations are valid:

$$\sum_{p|t=0} (y^p)^2 \approx \sum_{p|t=0} (y^p) \approx NP(y = 1, t = 0) \tag{9}$$

$$\sum_{p|t=0} (1 - y^p)^2 \approx \sum_{p|t=0} (1 - y^p) \approx NP(y = 0, t = 1) \tag{10}$$

with N the number of samples in the training set, and $P(y = 1, t = 0)$ and $P(y = 0, t = 1)$ the probabilities of incorrectly classifying a sample of resp. class 0 and class 1.

In this simple problem, two errors can be distinguished: incorrectly labelling of a sample as class 0 or as class 1. The squared error can be approximated using eq. 9 and eq. 10, by:

$$SQE \approx N\left(P(y = 1, t = 0) + P(y = 0, t = 1)\right) \tag{11}$$

which, except for the factor N, is equal to the average risk if the costs of both types of errors are set to 1. For this example, the error measure can easily be adjusted to incorporate different costs for the two different types of errors:

$$AVR \approx \lambda(1|0) \sum_{p|t=0} (y^p)^2 + \lambda(0|1) \sum_{p|t=1} (1 - y^p)^2 \tag{12}$$

with $\lambda(1|0)$ the cost of incorrect assignment of label 1 and $\lambda(0|1)$ the cost of incorrect assignment of label 0.

The resulting error measure is an approximation of the AVR *and* a continuous and differentiable function in the output y of the network.

Note: if the assumption that $y^p \approx 0$ or $y^p \approx 1$ is not valid, it can be shown that the squared error measure and the average risk for unity costs, although different do have the same minimum (provided that the training set is very large). This is also valid for the cost weighted squared error measure of eq.12 [2]. This follows from the fact that for large enough training sets and correct training the outputs of the neural network converge to the a posteriori conditional class probability density functions [2]. Choosing the output with the maximum value (or in networks with one output thresholding at 0.5) then results in the minimum error classifier (i.e. minimum average risk with unity costs).

Next consider the pulse detection problem described earlier in this paper. In this case three types of errors can be distinguished:

- false alarm
- missed pulse
- displacement over 1 period

The first error type (false alarm) occurs if the network output $y^p = 1$, and all the target outputs within a one period distance are zero:

$$t^{p-1} = 0 \wedge t^p = 0 \wedge t^{p+1} = 0, and\ y^p = 1 \tag{13}$$

The second error type (missed pulse) occurs if the target output $t^p = 1$, and all network outputs for the patterns within a one period distance are zero:

$$t^p = 1, and\ y^{p-1} = 0 \wedge y^p = 0 \wedge y^{p+1} = 0 \tag{14}$$

The third type of error (displacement) occurs if:

$$t^{p-1} = 1 \vee t^{p+1} = 1, and\ y^p = 1 \tag{15}$$

If it is assumed that there will always be at least two periods between successive pulses, the $\vee$ and $\wedge$ can be substituted by additions. The average risk can then be approximated in a similar way as in eq.12:

$$AVR \approx \lambda_1 \sum_{p|t^{p-1}+t^p+t^{p+1}=0} (y^p)^2 + \lambda_2 \sum_{p|t^p=1} (1 - y^{p-1} - y^p - y^{p+1})^2 + \lambda_3 \sum_{p|t^{p-1}+t^{p+1}=1} (1 - y^p)^2 \quad (16)$$

with $\lambda_1, \lambda_2, \lambda_3$ are the costs of the three types of errors.

As an alternative, the $\wedge$'s can be substituted by multiplications.
For step detection a similar optimisation criterion can be used, because the desired output is similar to that of pulse detection.

EXPERIMENTAL VERIFICATION

Two experiments were set up. A pulse detection experiment to demonstrate the validness of the proposed method and a step detection experiment in which the performance of the neural networks are also compared to other operators. Eq.16 and the alternative using multiplications in stead of additions were used as error measures.

Pulse detection

For the experiments with pulse detection, the cost of displacements was set to 0 and the costs of missed pulses and false alarms were both set to 1:

- $\lambda_1 = \lambda_2 = 1$
- $\lambda_3 = 0$

As a training set a signal similar to the one in fig.1 was used, i.e. pulses with some low frequency noise. In the target output signal, shifts of 1 period in random directions were artificially generated with a probability of $\frac{1}{3}$ for both directions left and right. The pulse amplitude of the input signal was random with a uniform distribution between 0.0 and 1.0. Noise, obtained by filtering Gaussian noise with amplitude 1.0 with a Gaussian filter with $\sigma_{psf} = 2.0$ was added to the signal. The training set contained 10000 patterns.
The network used for detection of pulses was a 3 layer network with 7 inputs, 1 output and 4 processing elements in the hidden layer. The network was trained with 5000 cycles through the training set. The learning rate was set to 0.001 to avoid problems with convergence, that sometimes occur for high learning rates. No momentum was used. For the initialisation of the weights random values with a uniform distribution between 0.0 and 0.1 were used. All evaluations were performed with independent evaluation sets with the same statistics as the training sets.
Fig.3 shows the results of the pulse detection experiment. Fig.3a shows the reference and input signal (The reference is given in a different scale for clarity; it ranges from 0 to 1). Clearly some of the pulses are shifted to the left and some to the right. Fig.3b shows the output of the SQE trained network. As expected, the network tends to suppress the shifted pulses. Both AVR trained networks (fig.3c and fig.3d) show a nearly perfect output. Fig.3e shows the squared error for each of the three networks. The SQE trained network (solid curve) has the lowest SQE and it is quite clear that the AVR trained networks do not attempt to minimise the SQE. Fig.3f shows the AVR of the networks during training. Both the AVR trained networks reach a value close to 0. The SQE trained network (solid curve) clearly performs worse according to the AVR error measure.
A control experiment without shifted pulses resulted in equal performance of AVR and SQE trained networks.

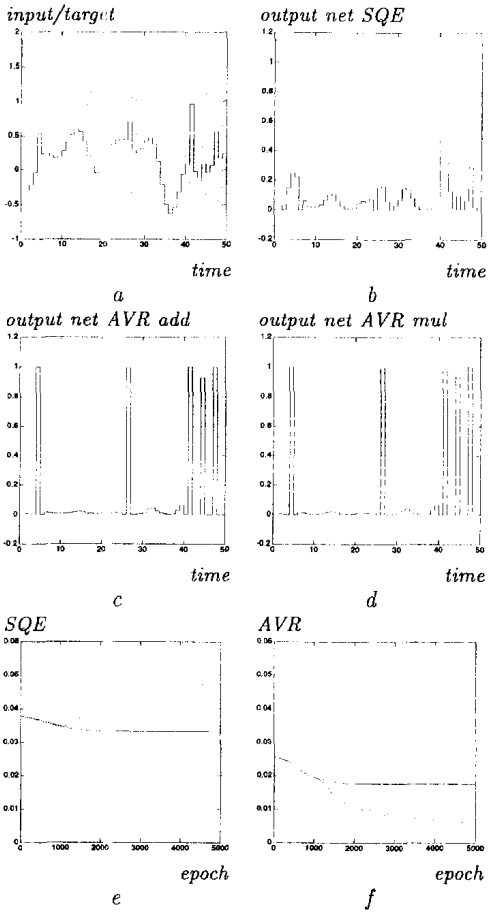

FIGURE 3: Pulse detection experiment results

Step detection

Fig.4 shows the input signal and target output signal of the step detection experiment. For this case the AVR error measure for training of the network was extended with error types for displacement over 3 periods and an improved definition of missed pulse and false alarm errors. Furthermore the input signal

was normalised before presentation to the networks. Two networks, one trained for minimum SQE and for minimum AVR were compared to 5 other step detector methods. The details of the step detectors and the evaluation can be found in [5].

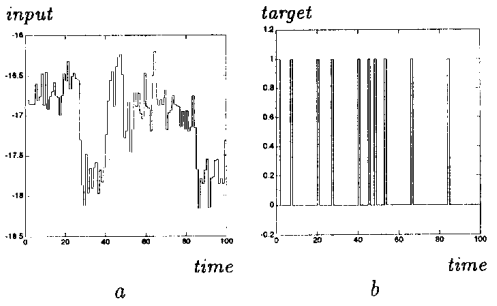

FIGURE 4: Input signal and target output for step detection experiment

The results of the comparison are shown in the table below. Note the extremely low missed step error of the AVR trained network, and the somewhat higher error for the displacements errors. This is exactly as it should be, because we set the cost of small displacement errors lower than for missed steps. The results show that with the AVR method neural networks can be designed that can very well compete with other detection operators.

operator	AVR	missed steps	false alarms	mult. detec.	displ. 1	displ. 2	displ. 3
NN, avr	.0226	.0174	.0016	.0005	.0076	.0008	.0002
Sarkar-Boyer	.0228	.0183	.0014	.0002	.0075	.0006	.0002
NN, sqe	.0232	.0179	.0017	.0007	.0066	.0006	.0002
Boie-Cox	.0233	.0189	.0015	.0001	.0079	.0005	.0001
Petrou-Kittler	.0235	.0186	.0015	.0001	.0088	.0007	.0002
Spacek	.0235	.0189	.0016	.0000	.0081	.0006	.0001
Shen-Castan	.0241	.0191	.0015	.0011	.0061	.0005	.0002

TABLE 1: Results of step detector comparison

CONCLUSIONS

It has been shown that under certain circumstances the sum of squared errors criterion (SQE), as is used in the standard back-propagation learning rule, does not always lead to a correct solution. This is e.g. the case if training patterns are not independent. The sum of squared errors cannot differentiate between the different types of errors that occur in these signals. Also the SQE does not take into account different costs of errors. The average risk (AVR), that does not have these defects, is proposed as an alternative error measure. An important advantage of the AVR is that it can be adjusted to a certain problem to take into account those errors that are specific for the problem and by assigning appropriate costs.
A method is proposed to use the AVR as an optimisation criterion in error backpropagation. The proposed method is demonstrated with networks for pulse and step detection.

References

[1] Barnard, E. ,1992, "Optimisation for training neural nets", *IEEE Trans. on Neural Networks*, vol.3, 1992, pp.232-240

[2] Devijver, P.A. ,1973, "Relationships between statistical risks and the least-mean-square-error design criterion in pattern recognition", *Proc. of the 1st Int. Joint Conf. on Pattern Recognition*, Washington D.C., 1973, pp.139-148

[3] Devijver, P.A., Kittler, J. ,1982, "Pattern recognition: a statistical approach", pp.22-26, New York, Prentice Hall, 1982

[4] Heijden, F. van der ,1992, "A statistical approach to edge and line detection in digital images", Dissertation, University of Twente, Netherlands, ISBN 90-9004922-3

[5] Heijden, F. van der, 1995, "Edge and line feature extraction based on covariance models", Pattern Recognition and Machine Intelligence, vol.17, no.1, 1995

[6] Holt, M.J.J. ,1992, "Comparison of generalization in multi-layer perceptrons with the log-likelihood and least-squares cost functions", *Proc. of the 11th IAPR Int. Conf. on Patt. Recognition*, Thue Hague, Aug.30 - Sept.3, 1992, vol.II, pp.17-20

[7] Hush, D.R., Salas, J.M. ,1988, "Improving the learning rate of back-propagation with the gradient reuse algorithm", *Proc. of the IEEE Int. Conf. on Neural Networks*, San Diego, California, July 24-27, 1988, vol.I, pp.441-447

[8] Kraaijveld, M.A. ,1993, "Small sample behavior of multi-layer feedforward network classifiers: theoretical and practical aspects", Dissertation, ISBN 90-6275-934-3/CIP

[9] Kung, S.Y., Brown, T.A. ,1988, "An algebraic projection analysis for optimal hidden units size and learning rates in backpropagation learning", *Int. Conf. on Neural Networks*, July 1988

[10] Nedeljkovic, V. ,1993) "A novel multilayer neural networks training algorithm that minimizes the probability of classification error", *IEEE Trans. on Neural Networks*, NN-4,4). pp.650-659, July 1993

[11] Parker, D.B. ,1987) "Optimal algorithms for adaptive networks: second order backpropagation, second order direct propagation, and second order hebbian learning", *Proc. of the 1st IEEE International Conf. on Neural Networks*, June, 1987, vol.II, pp.593-600

[12] Raudys, S.J., Jain, A.K. ,1991, "Small sample size problems in designing artificial neural networks", in: *Artificial Neural Networks and Statistical Pattern Recognition: Old and New Connections*, Sethi and Jain ,eds.), Elsevier Science Publishers, pp. 33-50, 1991

[13] Rumelhart, D.E., Hinton, G.E., Williams, R.J. ,1986, "Learning internal representations by error propagation", In: *Parulel distributed processing, explorations and the microstructures of cognition*, Rumelhard and McClelland ,Eds), pp.282-317, MIT Press

[14] Spreeuwers, L.J. ,1991, "A neural network edge detector", *Proc. of the SPIE/SPSE Symposium on Electrical Imaging Science & Technology, nonlinear image processing II*, San Jose, California, USA, 1991, SPIE vol.1451, pp.205-215

[15] Spreeuwers, L.J. ,1992, "Image filtering with neural networks. Applications and performance evaluation", Dissertation, University of Twente, Netherlands, ISBN 90-9005555-X

[16] Spreeuwers, L.J., Heijden, F. van der ,1992, "An edge detector evaluation method based on average risk", *Proc. of the 2nd International Workshop on Robust Computer Vision*, ISBN 3-87907-234-4, pp.79-89, Bonn, March 9-12, 1992

[17] Spreeuwers, L.J., Heijden, F. van der ,1992, "Evaluation of edge detectors using average risk", *11th IAPR International Conference on pattern recognition*, pp.771-774, The Hague, Netherlands, August 30 - September 3, 1992

[18] Wilkins, B.R., Ford, N.L. ,1972, "The analysis of training sets for adaptive pattern classifiers", in *Machine Perception of Patterns and Pictures, Inst. Physics, Conf. Ser.*, 13:267, 1972