ELSEVIER

Pattern Recognition Letters 16 (1995) 1221-1236

Pattern Recognition Letters

Training neural networks for minimum average risk with a special application to context dependent learning

Luuk Spreeuwers *

Department of Electrical Engineering, University of Twente, P.O. Box 217, 7500 AE Enschede, Netherlands

Received 15 December 1992; revised 24 July 1995

Abstract

The least squares criterion, as used by the backpropagation learning rule in multi-layer feed forward neural networks, does not always yield a solution that is in accordance with the desired behaviour of the neural network. This is for example the case when differentiation between different types of errors is required and the costs of the error types must be taken into account. In this paper the application of other error measures, specifically matched to the application, is investigated. The error measures used are based on the average risk, a function that is a weighted combination of the probabilities on the different types of errors that may occur. Special attention is payed to applications where the input patterns are not independent, and the average risk does not depend on the output of a single input pattern, but on its neighbourhood, or context. The ideas are illustrated with pulse detection in a one dimensional signal.

Keywords: Error backpropagation learning; Optimisation criterion; Average risk; Context learning

1. Introduction

Multi-layer feed forward neural networks using the error backpropagation learning rule have been proven successful in numerous applications. Therefore, they have become a popular subject for applicative and theoretic research. The ability of these networks to realize complex functions, combined with the learning by example approach, makes these networks well-suited to many different applications in the fields of pattern recognition, control, image processing etc.

The error backpropagation learning rule, as described by Rumelhart, Hinton and Williams (1986) uses the squared error (SQE) as an error measure. The error backpropagation learning rule is an optimisation method that is used to minimize this error measure. In many cases the application of the least squares criterion results in optimal or acceptable suboptimal solutions. There are cases, however, where the optimal solution according to the least squares criterion does not coincide with the desired solution, because it does not take into acount the specific types of errors for the application. This research was initiated because just that problem was encountered in the design of a neural network for edge detection in grey level images (Spreeuwers, 1991; Spreeuwers, 1992). The performance measure we use for the evaluation of edge detectors, the average risk (AVR) evaluation method (van der Heijden, 1992; Spreeuwers and van der Heijden, 1992a; Spreeuwers and van der Heijden, 1992b), takes into account different types of errors and weighs them in accordance with the application. For example, if the objective of an image processing

[1] Email: luuk@mi.el.utwente.nl

SSDI 0167-8655(95)00086-0

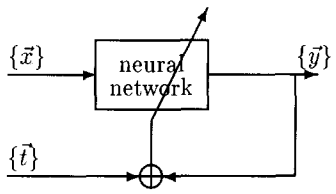

Fig. 1. Back-propagation learning; the error of the output is minimised using an iterative optimisation strategy. The error is a difference measure between the actual and target outputs.

system is to recognize 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 penalize these displacement errors very heavily, as will be explained in detail later. In this case we are looking for operators that are optimised for minimum AVR, rather than for minimum SQE. It will be shown that the SQE measure is a special case of the AVR error measure.

2. Accuracy of neural network solutions

The objective of training a neural network is to obtain a function with a certain desired behaviour. For this purpose, a certain size and architecture are chosen for the network, and the connection weights are adjusted until the network approximates the desired behaviour as accurately as possible. The error backpropagation learning rule is an iterative numerical function optimisation method that is used for this task. The networks are trained using examples of the desired operation, i.e. input patterns with the desired corresponding output patterns. The learning rule attempts to minimize the error of the network for the example set as a whole. This process is illustrated in Fig. 1. The accuracy of the approximation of the resulting behaviour to the desired behaviour depends on the following factors:

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

Since the desired behaviour is actually described by the training set, this determines the performance of the resulting network to a great extent. The training set should be representative of the desired behaviour. This means that enough training samples of good quality must be available to represent the desired behaviour. Literature addressing the dependency of neural network (and other operators) performance on the training set is e.g. (Koplowitz and Brown, 1981; Kraaijveld, 1993; Raudys and Jain, 1991).

The neural network must also be able to approximate the desired behaviour sufficiently accurately. If the desired behaviour becomes more complex, more processing elements and/or layers are required. However, if the network becomes very large, this slows down the learning process, and it may result in bad generalization from the training data. One paper addressing the problem of choosing the optimal size for the network for a certain application is (Kung and Hwang, 1988).

Less attention in literature is given to the optimisation criterion used in error backpropagation neural networks. The 'standard' error backpropagation learning rule uses the least squares error criterion, which minimizes the sum of the squared errors of the training set:

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

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

The optimal solution with respect to the least squares error criterion, is the set of weights for a network that results in the smallest SQE, for the given training set and network. This solution, however, does not always correspond to the desired solution, as was e.g. the case for the neural network edge detector mentioned in the introduction. In (Holt, 1992; Holt and Semmani, 1990) 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 (Nedeljkovic, 1992; Nedeljkovic, 1993) an error measure is proposed that minimises the probability of classifica-

tion errors, rather than the SQE. Neither of these approaches introduces a weighted cost measure of different types of errors like the AVR. Because of this cost weighting of different error types, the AVR can be better adapted to a specific application with its specific types of errors.

The error backpropagation learning rule is a gradient descent optimisation method. This means that the weights of the network are always adjusted in the direction of the negative derivative of the SQE. It should be remarked that this optimisation method is not very robust: it can easily get stuck in local minima (gradient = 0), and learning can be very slow if the gradient is small. Some enhancements to error backpropagation and some other optimisation methods for feed-forward neural networks can be found in (Barnard, 1992; Dahl, 1987; Herz et al., 1991; Hush and Salas, 1988; Jacobs, 1988; Parker, 1987).

In this paper, the influence of the error criterion on the performance of neural networks is investigated. In the next sections first it is analysed under what conditions the least squares error criterion does not result in a correct solution and then it is shown how it can be replaced by a more appropriate optimisation criterion. In Section 5 an error criterion, the minimum average risk, is described that takes into account the costs of different types of errors. 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.

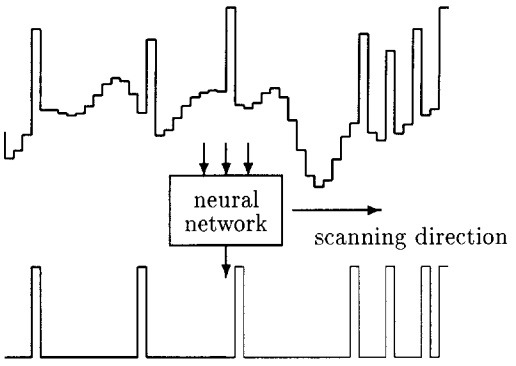

Fig. 2. Pulse detection with a neural network. The network scans the signal, taking a window of the signal as its input. Its single output should be 1 if at the centre of the window a pulse is present and 0 otherwise.

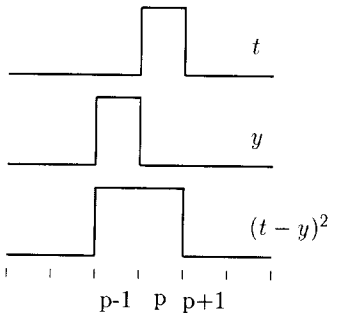

Fig. 3. Double counting of errors for shifted detected pulse relative to target pulse. The target at time p is shifted to the right relative to the network output. Because the squared error does not differentiate between error types, these errors are counted double, whereas in reality they are often less serious than other types of errors.

3. Non-optimal least squares solutions

If the SQE measure for a network and a training set is exactly equal to 0, then for each training sample the network generates exactly the target output. Thus if $SQE = 0$, then the behaviour, as described by the training set, is optimally realized in the neural network. This, however, is not necessarily equivalent to the desired behaviour. For instance if the training set does not represent the desired behaviour well, the performance on the training set may be (nearly) perfect, but on another independent data set the performance may be much worse. For example a network with too many processing elements and connection weights may be tuned to specific training samples, which often results in worse generalisation.

There is another cause for discrepancy between the desired behaviour and the performance of a network on a training set. 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 to or too insensitive to certain types of errors. The problem and solution presented in this paper refer to these types of inaccuracies 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 sig-

nal 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 Fig. 2). In order to determine 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 temporal. 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. 3 the target output 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 for input pattern $p-1$ is 1 and for p it is 0. As shown in Fig. 3 this results in a contribution of 2 to the total squared error for these two input patterns. 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. As a matter of fact, the behaviour of a SQE minimised neural network can be predicted in this case. Suppose that the probability on a shift of one period to the left is $f(-1)$, a shift to the right $f(1)$ and no shift is $f(0)$. The outputs at position -1, 0 and $+1$ are: y_{-1}, y_0 and y_{+1}. The expectancy of the squared error for a pulse at the input will be:

$$\begin{aligned} E\{SQE\} = {} & f(-1)[(1-y_{-1})^2 + y_0^2 + y_{+1}^2] \\ & + f(0)[y_{-1}^2 + (1-y_0)^2 + y_{+1}^2] \\ & + f(+1)[y_{-1}^2 + y_0^2 + (1-y_{+1})^2]. \end{aligned} \quad (2)$$

The minimum squared error occurs if the derivatives of SQE to y_{-1}, y_0 and y_{+1} are equal to 0 thus:

$$\frac{\partial E\{SQE\}}{\partial y_{-1}} = \frac{\partial E\{SQE\}}{\partial y_0} = \frac{\partial E\{SQE\}}{\partial y_{+1}} = 0 \quad (3)$$

$$\begin{aligned} -2f(-1)(1-y_{-1}) + 2f(0)y_{-1} + 2f(+1)y_{-1} &= 0, \\ 2f(-1)y_0 - 2f(0)(1-y_0) + 2f(+1)y_0 &= 0, \\ 2f(-1)y_{+1} + 2f(0)y_{+1} + 2f(+1)(1-y_{+1}) &= 0 \end{aligned} \quad (4)$$

which leads to:

$$\begin{aligned} y_{-1} &= \frac{f(-1)}{f(-1)+f(0)+f(+1)}, \\ y_0 &= \frac{f(0)}{f(-1)+f(0)+f(+1)}, \\ y_{+1} &= \frac{f(+1)}{f(-1)+f(0)+f(+1)}. \end{aligned} \quad (5)$$

Thus if e.g. $f(-1) = f(0) = f(+1) = \frac{1}{3}$, then $y_{-1} = y_0 = y_{+1} = \frac{1}{3}$ also.

Because the target output is 1, the error is quite high and will never be close to 0. Furthermore, if there is noise present in the signal, detection of the pulses becomes very difficult, because the output is so weak. 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 types (generally called *costs*). The application of the SQE error measure in this light is rather limited.

4. Use of other optimisation criteria in error backpropagation

The derivation of the error backpropagation learning rule by Rumelhart et al. (1986), begins with the introduction of the error measure, which is almost equal to the the least squares error criterion. The error criterion they use is defined as:

$$E = \frac{1}{2}\sum_p \sum_i (t^{pi} - y^{pi})^2 \quad (6)$$

where:
t^{pi} = the target output for input pattern x^p, output i;
y^{pi} = the actual output for input pattern x^p, output i;

the summations are over all outputs i and training samples p.

For a multi-layer feed forward network Rumelhart et al. (1986) derived the following weight update rule for a gradient descent optimisation method:

$$\Delta w_k^{pij} = \eta \delta_k^{pi} o_{k-1}^{pi} \tag{7}$$

where:

Δw_k^{pij} = the weight update of a connection between processing element j in layer $k-1$ to processing element i in layer k for an input pattern x^p;

η = the learning rate;

δ_k^{pi} = an error measure local to the processing element i in layer k for input pattern x^p;

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. (7) are determined recursively from the output in the direction of the input (hence the name error backpropagation). For the output layer (layer L) the δ's can be calculated directly from the error $t^{pi} - y^{pi}$:

$$\delta_L^{pi} = (t^{pi} - y^{pi}) f'(net_L^{pi}). \tag{8}$$

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

$$\delta_k^{pi} = \sum_h \delta_{k+1}^{pi} w_{k+1}^{hi} f'(net_k^{pi}) \tag{9}$$

where:

$f'()$ = 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;

w_{k+1}^{hi} = the connection weight of the connection between processing element i in layer k to processing element h in layer $k+1$;

summation is over all processing elements h in layer $k+1$.

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

The question now is how these equations change if an error measure other than the SQE of Eq. (6) 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 the SQE error measure of Eq. (6) is not substituted into the calculation of the δ's of the output layer, instead of Eq. (8) the following expression is obtained:

$$\delta_L^{pi} = -\frac{\partial E}{\partial o_L^{pi}} f'(net_L^{pi}). \tag{10}$$

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.

5. Average risk, an error measure matched to the application

In order to obtain an error measure that can be used to evaluate the performance of an operator for a certain application, the types of errors that can occur should be analysed. Furthermore the seriousness of each type of error should be investigated or defined and the probability of occurrence of the errors must be determined. The latter can be estimated by the frequency of occurrence of each type of error in the training set:

$$\hat{P}(\varepsilon_i) = \frac{N(\varepsilon_i)}{N} \tag{11}$$

where:

$\hat{P}(\varepsilon_i)$ = the estimated probability (frequency) of error type ε_i;

$N(\varepsilon_i)$ = the number of occurrences of error type ε_i in the training set;

N = the number of samples in the training set.

The average risk (AVR) (Devijver and Kittler, 1982; Spreeuwers and van der Heijden, 1992a; Spreeuwers and van der Heijden, 1992b) is an error measure that takes into account these three factors: 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 average risks of the individual errors:

$$AVR = \sum_i P(\varepsilon_i) \lambda(\varepsilon_i) \tag{12}$$

where:

$P(\varepsilon_i)$ = the probability of error type ε_i;

$\lambda(\varepsilon_i)$ = the cost of an error type ε_i;

summation is over all error types i.

The probabilities of the different error types can be estimated from a training set using Eq. (11). Thus, for a given set of costs $\boldsymbol{\lambda}$, 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. Incidentally, for classification this leads to the Bayes decision rule, which is generally accepted as *the* optimal classifier (see e.g. (Devijver and Kittler, 1982)).

From Eq. (10) 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. (12) is neither continuous nor differentiable, in order to train a neural network for minimum average risk, an approximation of Eq. (12) must be derived that is continuous and differentiable. The solution to this problem, presented below, is based on 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 two-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 for a certain training set of this network is:

$$SQE = \sum_p (t^p - y^p)^2. \tag{13}$$

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

$$SQE = \sum_{p|t=0} (y^p)^2 + \sum_{p|t=1} (1 - y^p)^2. \tag{14}$$

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

$$\sum_{p|t=0} (y^p)^2 \approx \sum_{p|t=0} (y^p) \approx N(y = 1, t = 0) \approx N P(y = 1, t = 0) \tag{15}$$

$$\sum_{p|t=0} (1 - y^p)^2 \approx \sum_{p|t=0} (1 - y^p) \approx N(y = 0, t = 1) \approx N P(y = 0, t = 1) \tag{16}$$

where:

$N(y = 1, t = 0)$ = the number of incorrectly classified samples of class 0;

$N(y = 0, t = 1)$ = the number of incorrectly classified samples of class 1;

N = the number of samples in the training set;

$P(y = 1, t = 0)$ = the probability of incorrectly classifying a sample of class 0;

$P(y = 0, t = 1)$ = the probability of incorrectly classifying a sample of 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. (15) and Eq. (16), by:

$$SQE \approx N\,(P(y = 1, t = 0) + P(y = 0, t = 1)) \tag{17}$$

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:

$$AVR \approx \lambda(y = 1, t = 0) \sum_{p|t=0} (y^p)^2 + \lambda(y = 0, t = 1) \sum_{p|t=1} (1 - y^p)^2 \tag{18}$$

where:

$\lambda(y = 1, t = 0)$ = the cost of incorrect assignment of label 1;

$\lambda(y = 0, t = 1)$ = 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 sufficiently representative for the desired behaviour). This is also valid for the cost weighted squared error measure of Eq. (18) (Devijver, 1973). This follows from the fact that for a sufficiently representative training set and correct training the outputs of the neural network converge to the a posteriori conditional class probability density functions (Devijver, 1973; Spreeuwers, 1995). Choosing the output with the maximum value (or, for problems with 2 classes,

in networks with a single output, thresholding at 0.5) then results in the minimum error classifier (i.e. minimum average risk with unity costs) (Devijver and Kittler, 1982; Spreeuwers, 1992).

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

- the network detects a pulse but in reality there is no pulse;
- the network does not detect a pulse, but in reality there is one;
- the network detects a pulse, but displaced over one period relative to the target.

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:

$$(t^{p-1} = 0) \wedge (t^p = 0) \wedge (t^{p+1} = 0) \text{ and } (y^p = 1). \tag{19}$$

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:

$$(t^p = 1) \text{ and } (y^{p-1} = 0) \wedge (y^p = 0) \wedge (y^{p+1} = 0). \tag{20}$$

The third type of error (displacement) occurs if:

$$(t^{p-1} = 1) \vee (t^{p+1} = 1) \text{ and } (y^p = 1). \tag{21}$$

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. (18):

$$\begin{aligned} AVR \approx \lambda_1 & \sum_{p \mid t^{p-1}+t^p+t^{p+1}=0} (y^p)^2 \\ & + \lambda_2 \sum_{p \mid t^p=1} (1 - y^{p-1} - y^p - y^{p+1})^2 \\ & + \lambda_3 \sum_{p \mid t^{p-1}+t^{p+1}=1} (1 - y^p)^2 \end{aligned} \tag{22}$$

where:
$\lambda_1, \lambda_2, \lambda_3$ the costs of the three types of errors.

As an alternative, the $\wedge$'s can be substituted by multiplications. The approximation for the AVR then becomes:

$$\begin{aligned} AVR \approx \lambda_1 & \sum_{p \mid t^{p-1}+t^p+t^{p+1}=0} (y^p)^2 \\ & + \lambda_2 \sum_{p \mid t^p=1} \left((1 - y^{p-1})(1 - y^p)(1 - y^{p+1})\right)^2 \\ & + \lambda_3 \sum_{p \mid t^{p-1}+t^{p+1}=1} (1 - y^p)^2. \end{aligned} \tag{23}$$

The difference between the two error measures of Eq. (22) and Eq. (23) is in the way missed pulses are detected. In Eq. (22) the sum of the output for three sequential periods is used for the estimation of the missed pulse error. This only works correctly if there are no multiple responses on a single input. For example, if the sum of the output for three successive periods is close to 1 (e.g. $t^p = 1$ *and* $y^{p-1} = y^p = y^{p+1} = \frac{1}{3}$), this may result in a lower missed pulse error (in the example it is even 0). On the other hand, if for three successive periods the output is high, so that the sum is greater than 1 (e.g. $t^p = 1$ *and* $y^{p-1} = y^p = y^{p+1} = 1$), the second term of Eq. (22) gives rise to a missed pulse error that in fact is incorrect.

Eq. (23) handles the missed errors more correctly. For multiple responses the missed pulse error (2nd term of Eq. (23)) becomes smaller but it cannot disappear like in Eq. (22), nor do multiple responses with high output give rise to incorrect missed pulse errors.

In the beginning of the learning process the output generally is quite low. When it starts to grow, the first situation (i.e. the sum of the output for three successive periods is close to 1) may occur and the missed error is suppressed. Therefore in the beginning of the learning process, just after the outputs have started to grow, one could expect that in some cases Eq. (23) would result in faster convergence. The other effect, incorrect detection of missed pulses by Eq. (22) may result in the reverse situation: because the error is larger, the rate of convergence might increase.

Normally both error measures should converge to the same minimum, however.

6. Experimental verification

6.1. Description of the experiments

In order to test the validity of the proposed method, a number of experiments for the pulse detection prob-

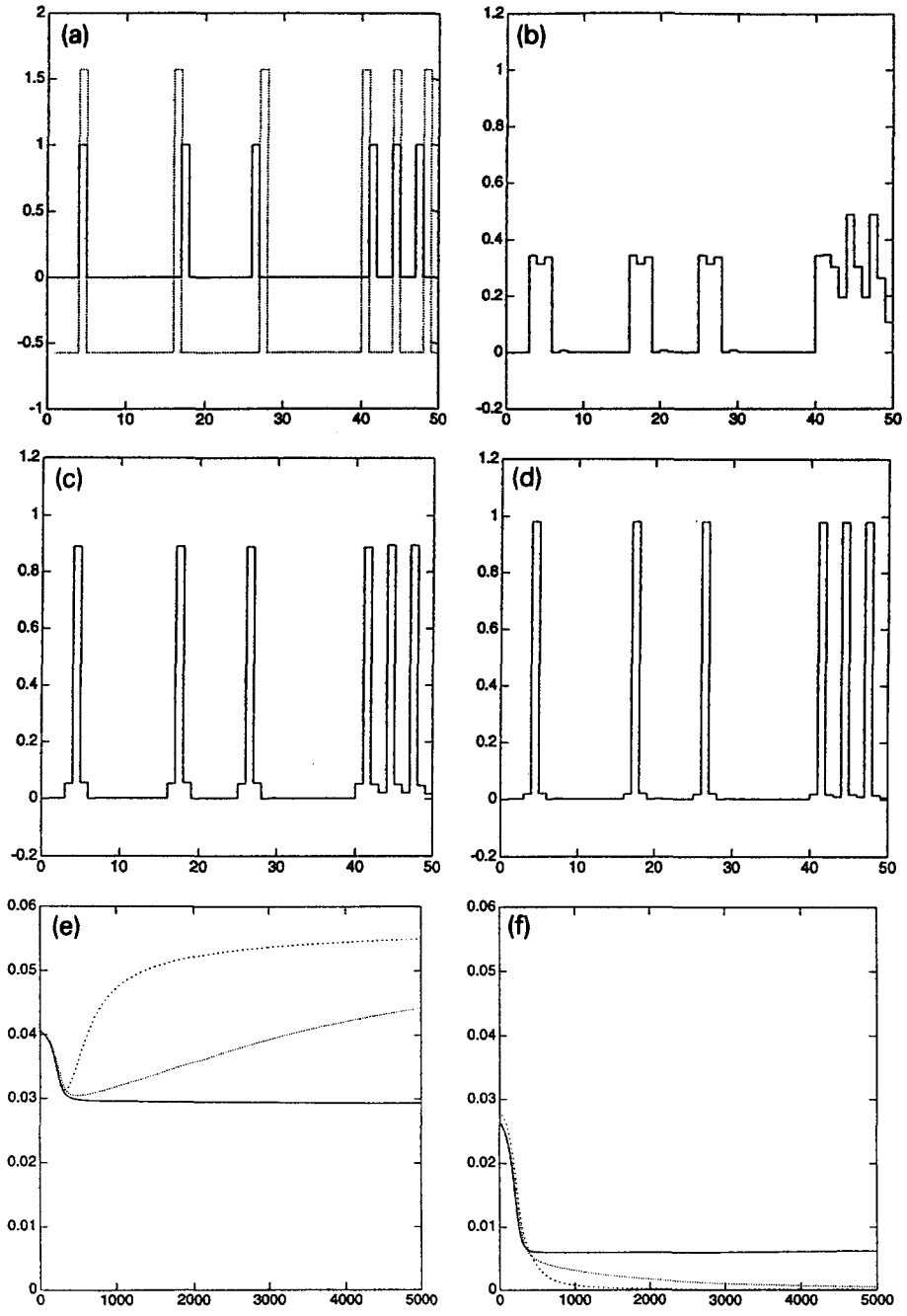

Fig. 4. Results of pulse detection experiment: probability on shift left and right is 0.33; pulse amplitude is 1.0; no noise. (a) Network input (solid) and reference (dotted). (b) SQE trained network output. (c) AVR1 trained network output. (d) AVR2 trained network output. (e) Squared error vs. learn cycles for SQE (solid), AVR1 (dotted) and AVR2 (dashed) trained networks. (f) AVR vs. learn cycles for SQE (solid), AVR1 (dotted) and AVR2 (dashed) trained networks.

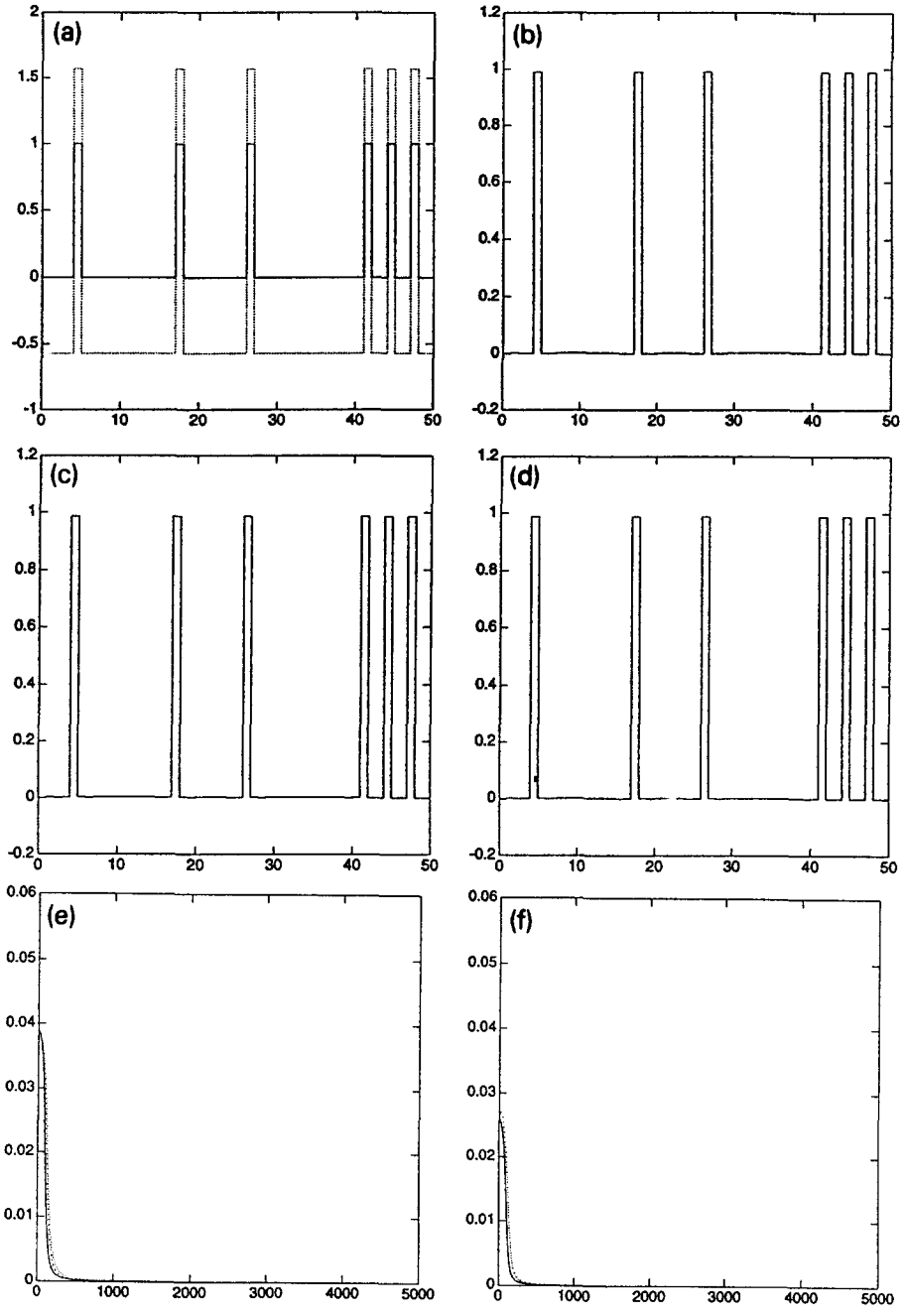

Fig. 5. Results of pulse detection experiment: no shifts; pulse amplitude is 1.0; no noise. (a) Network input (solid) and reference (dotted). (b) SQE trained network output. (c) AVR1 trained network output. (d) AVR2 trained network output. (e) Squared error vs. learn cycles for SQE (solid), AVR1 (dotted) and AVR2 (dashed) trained networks. (f) AVR vs. learn cycles for SQE (solid), AVR1 (dotted) and AVR2 (dashed) trained networks.

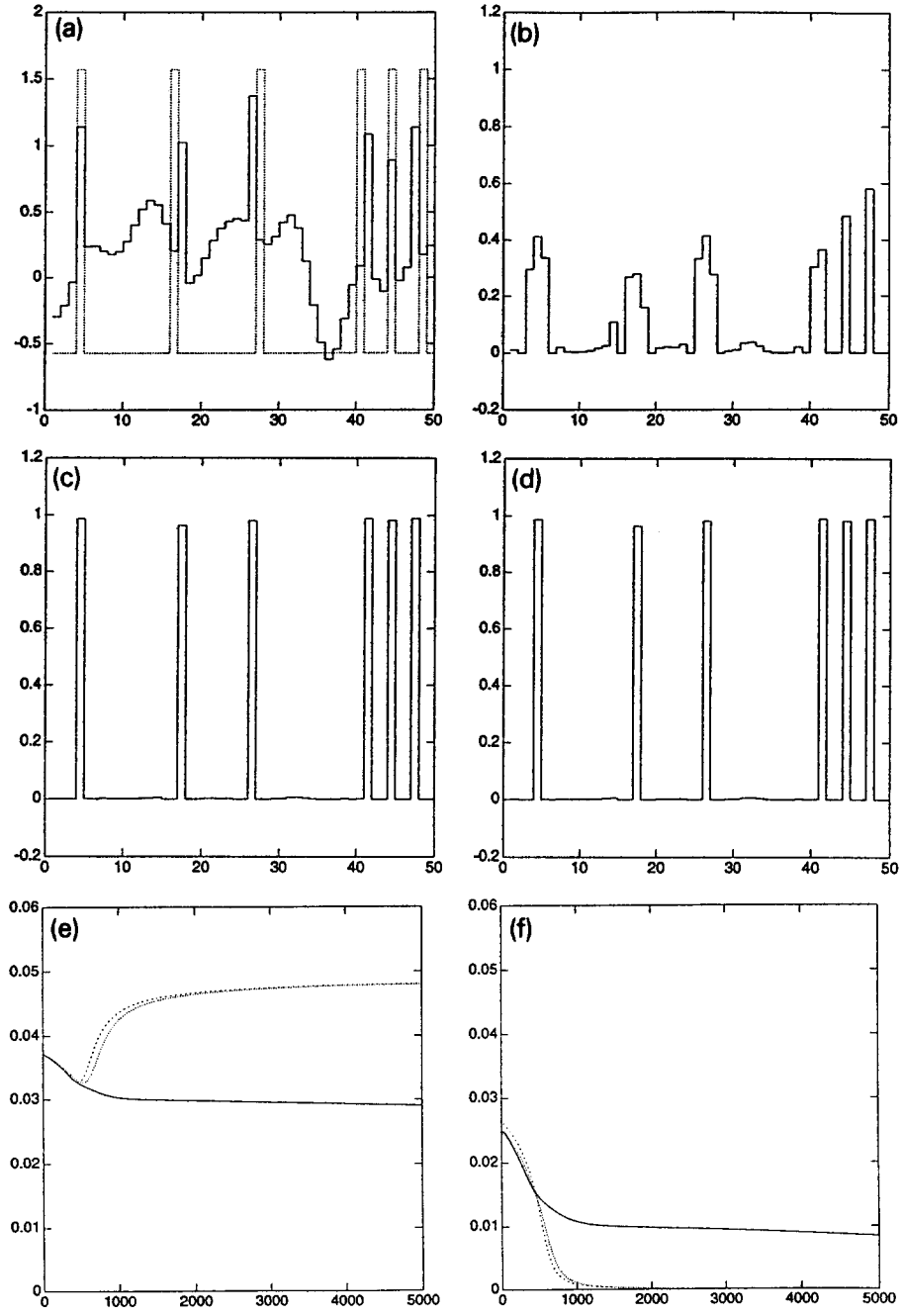

Fig. 6. Results of pulse detection experiment: probability on shift left and right is 0.33; pulse amplitude is 1.0; noise amplitude = 1.0; noise filter σ_{psf} is 2.0. (a) Network input (solid) and reference (dotted). (b) SQE trained network output. (c) AVR1 trained network output. (d) AVR2 trained network output. (e) Squared error vs. learn cycles for SQE (solid), AVR1 (dotted) and AVR2 (dashed) trained networks. (f) AVR vs. learn cycles for SQE (solid), AVR1 (dotted) and AVR2 (dashed) trained networks.

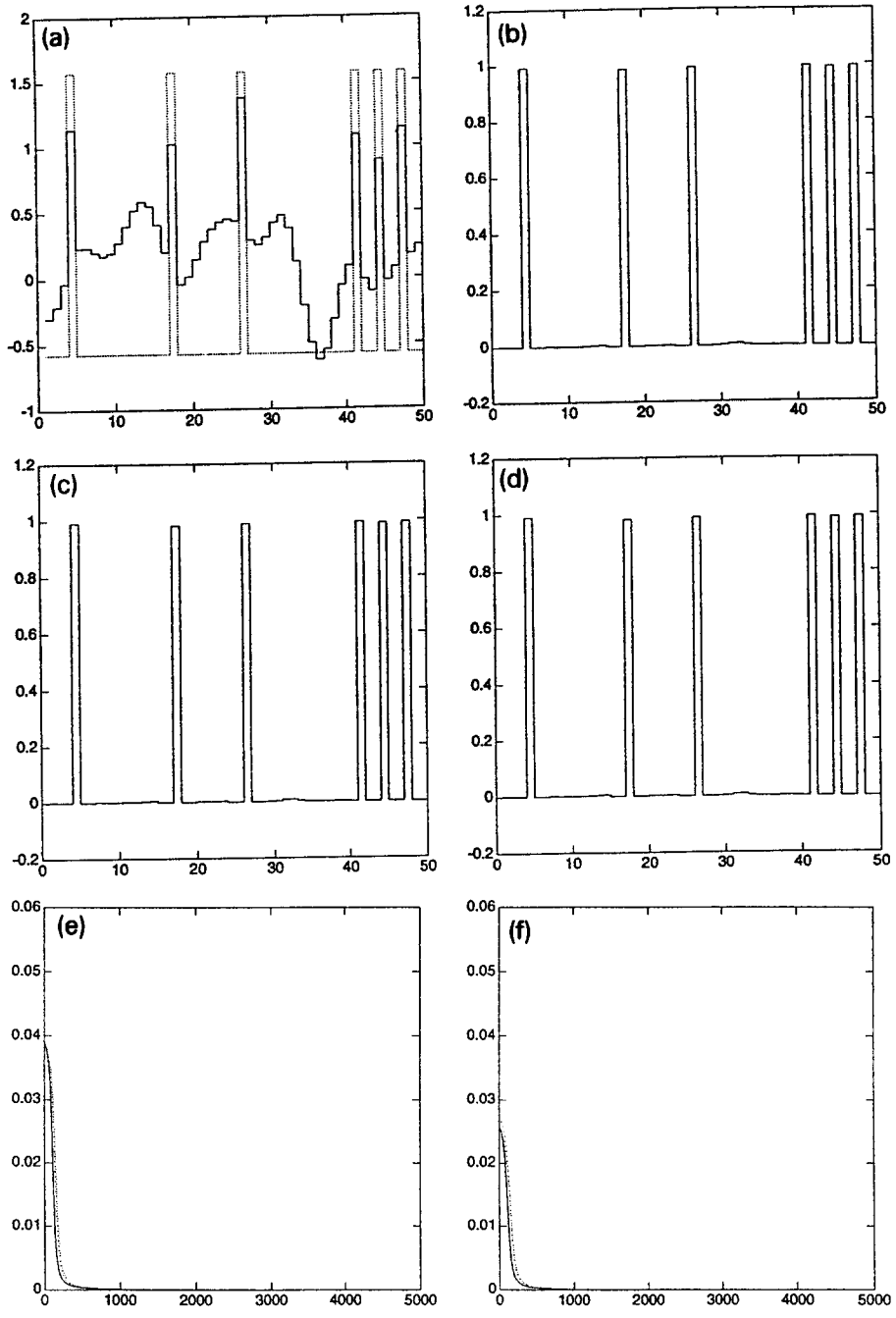

Fig. 7. Results of pulse detection experiment: no shifts; pulse amplitude is 1.0; noise amplitude = 1.0; noise filter σ_{psf} is 2.0. (a) Network input (solid) and reference (dotted). (b) SQE trained network output. (c) AVR1 trained network output. (d) AVR2 trained network output. (e) Squared error vs. learn cycles for SQE (solid), AVR1 (dotted) and AVR2 (dashed) trained networks. (f) AVR vs. learn cycles for SQE (solid), AVR1 (dotted) and AVR2 (dashed) trained networks.

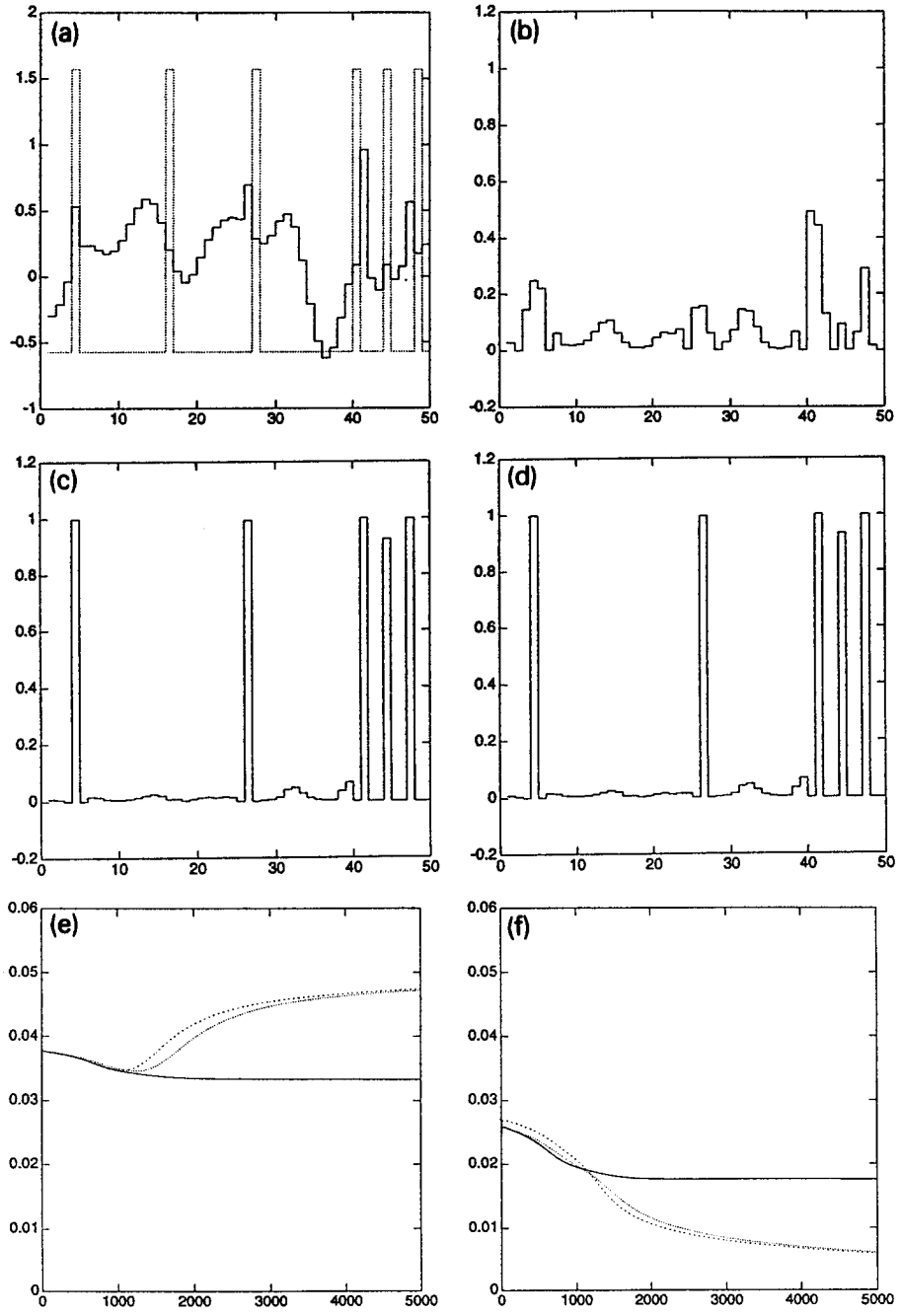

Fig. 8. Results of pulse detection experiment: probability on shift left and right is 0.33; pulse amplitude is uniformly distributed between 0.0 and 1.0; noise amplitude = 1.0; noise filter σ_{psf} is 2.0. (a) Network input (solid) and reference (dotted). (b) SQE trained network output. (c) AVR1 trained network output. (d) AVR2 trained network output. (e) Squared error vs. learn cycles for SQE (solid), AVR1 (dotted) and AVR2 (dashed) trained networks. (f) AVR vs. learn cycles for SQE (solid), AVR1 (dotted) and AVR2 (dashed) trained networks.

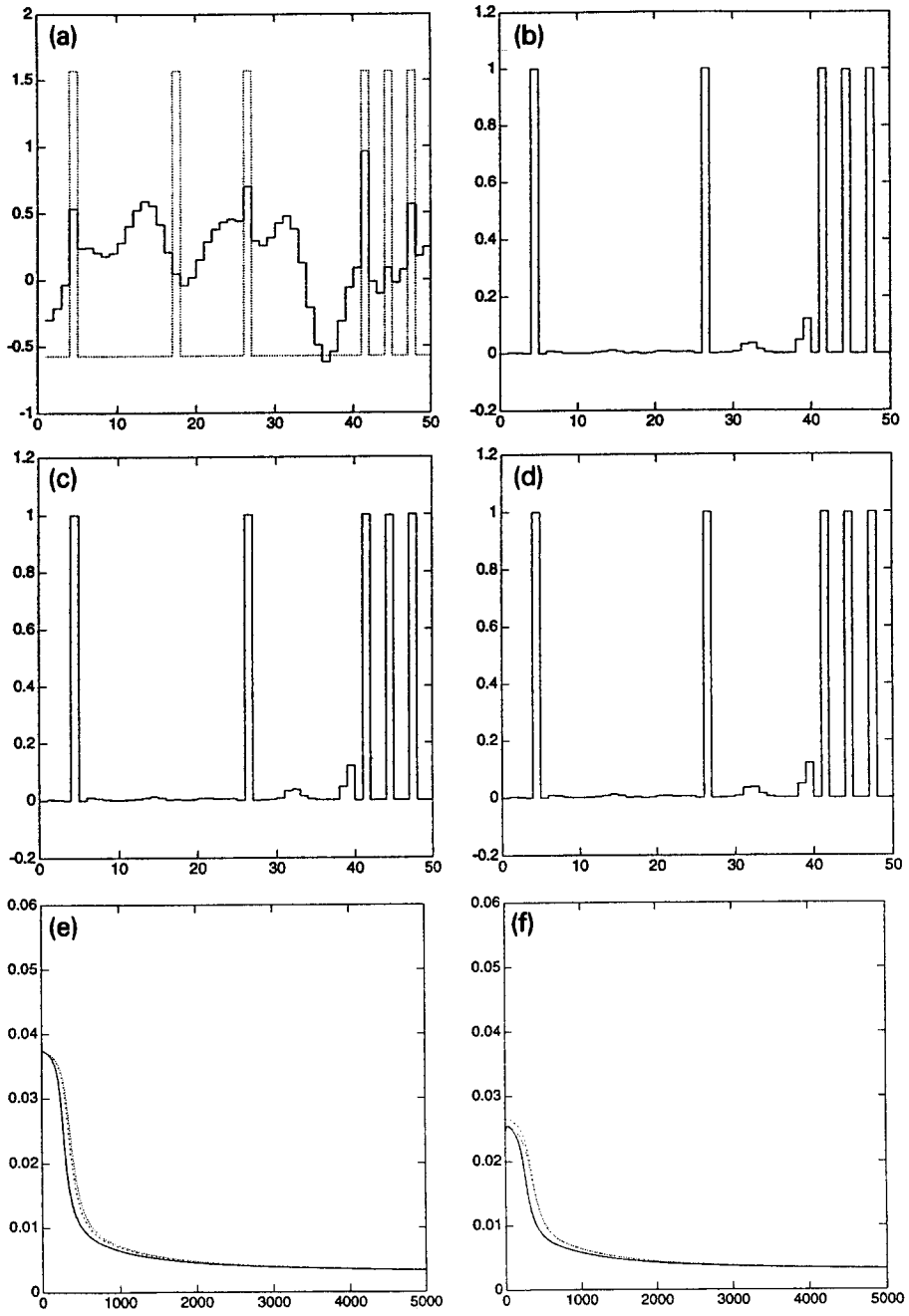

Fig. 9. Results of pulse detection experiment: no shifts; pulse amplitude is uniformly distributed between 0.0 and 1.0; noise amplitude = 1.0; noise filter σ_{psf} is 2.0. (a) Network input (solid) and reference (dotted). (b) SQE trained network output. (c) AVR1 trained network output. (d) AVR2 trained network output. (e) Squared error vs. learn cycles for SQE (solid), AVR1 (dotted) and AVR2 (dashed) trained networks. (f) AVR vs. learn cycles for SQE (solid), AVR1 (dotted) and AVR2 (dashed) trained networks.

lem were set up. Eq. (22) and Eq. (23) were used as error measures. For these experiments, 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. 2 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. A total of 3 experiments was carried out. Each experiment was repeated without shifted pulses.

The experiments carried out were:

Fig. 4 probability on shift left/right is $\frac{1}{3}$; no noise; fixed pulse amplitude=1.0
Fig. 5 probability on shift left/right is 0; no noise; fixed pulse amplitude=1.0
Fig. 6 probability on shift left/right is $\frac{1}{3}$; additive noise; fixed pulse amplitude=1.0
Fig. 7 probability on shift left/right is 0; additive noise; fixed pulse amplitude=1.0
Fig. 8 probability on shift left/right is $\frac{1}{3}$; additive noise; pulse amplitude random uniform between 0.0 and 1.0
Fig. 9 probability on shift left/right is 0; additive noise; pulse amplitude random uniform between 0.0 and 1.0

The noise in the experiments was obtained by filtering Gaussian noise with amplitude 1.0 with a Gaussian filter with $\sigma_{psf} = 2.0$. For each experiment the training set contained 10000 patterns. In the figures successively the following graphs are shown (for the signals (a), (b), (c), (d) only the first fifty samples of the 10000 are shown):

(a) the input signal (solid) and target output signal (dotted);
(b) the output signal of the SQE trained network;
(c) the output signal of the network trained with the error criterion of Eq. (22);
(d) the output signal of the network trained with the error criterion of Eq. (23);
(e) the SQE error during training of the SQE trained network (solid), the network optimised for AVR according to Eq. (22) (dotted) and the network optimised for the AVR according to Eq. (23) (dashed);
(f) the AVR during training of the SQE trained network (solid), the network optimised for AVR according to Eq. (22) (dotted) and the network optimised for the AVR according to Eq. (23) (dashed).

Note 1. For clarity and because it has a different amplitude range, in Figs. 4(a)–9(a) the input signal is given in a different scale than the other signals.

Note 2. The AVR and SQE in the graphs (e) and (f) are divided by the number of training samples in the training set.

The network used for the detection of pulses is a network with three layers: an input buffer layer with 7 inputs, a hidden layer with 4 processing elements and an output layer with a single element. The network was trained with 5000 cycles through the training set (5000 epochs). The learning rate was set to 0.001 to avoid problems with convergence, that sometimes occur for high learning rates. No momentum term was used. For the initialisation of the networks weights random values with a uniform distribution between 0.0 and 0.1 were used. The networks were operated in the so-called parallel training mode, which means the errors are accumulated and the weights are updated only after the whole training set has been processed once. Similar results were, however, obtained with sequentially trained networks. All evaluations were performed with independent evaluation sets with the same statistics as the training sets.

6.2. *Discussion of results*

The first experiment (Fig. 4) shows the results for fixed pulse amplitude and shifted pulses. Fig. 4(a) shows the reference and input signal. Clearly some of the pulses are shifted to the left and some to the right. Fig. 4(b) shows the output of the SQE trained network. As predicted in Section 3, the output is threefold for each pulse and the amplitude is about $\frac{1}{3}$. Both AVR-trained networks (Figs. 4(c) and 4(d)) show a nearly perfect output. Fig. 4(e) shows the squared error for each of the three networks. The network that was trained for minimum SQE (solid curve) has the

lowest SQE and it is quite clear that the networks that were trained for minimum AVR do not attempt to minimise the SQE. Fig. 4(f) shows the AVR of the networks during training. Both the networks that were trained for minimum AVR reach a value of almost 0. The network trained for Eq. (22) appears to converge slower than the network that minimises Eq. (23). As described in Section 5, this is probably caused by the suppression of the missed pulse errors of Eq. (22) during the first phase of the training process. The network trained for minimum SQE clearly performs worse according to the AVR error measure.

Fig. 5 shows the results of the same experiment but without the shifts. All three networks now show almost equal behaviour.

In Fig. 6 and Fig. 7 the results of the experiments with additive noise are given. As expected the network trained for minimum SQE is more sensitive to the noise if there are displacement errors. The networks trained for minimum AVR almost perfectly suppress the noise in this case. If there are no displacement errors (Fig. 7), the networks show equal performance.

In the last experiment (Fig. 8) the amplitude of the pulses is random with a uniform distribution between 0 and 1. In this case the difference between the performance of the networks trained for minimum SQE and AVR is even more apparent. Again for the signal without pulses (Fig. 9) the three networks perform nearly the same.

Note that the SQE of the networks trained for minimum AVR does not decrease monotonically in the case that there are displacement errors (Figs. 4(e), 6(e), 8(e)). The SQE even increases during the training process. However, in these cases the SQE is not a reliable measure of the effectiveness of the training phase. The AVR, which is adapted specifically to the types of errors that occur, does decrease monotonically.

6.3. Conclusions

The results of the experiments accurately follow the predictions of Section 3. Therefore it is likely that for problems like these, where the training vectors are not independent, and/or where a cost weighted error criterion is required, using an AVR-based error criterion for a neural network instead of the SQE will give a significant improvement in performance.

7. Conclusions

In this paper it is shown that under certain circumstances the sum of squared errors criterion (SQE), as is used in the standard backpropagation 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) 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 assign costs to the errors in accordance with the demands of the user.

It is shown how the error backpropagation learning rule can be adjusted to work with other error measures and an example is given of how to use the AVR with the error backpropagation learning rule. This involves finding an approximation for the AVR that is both continuous and differentiable. The proposed approach is demonstrated with an example of a pulse detection problem, where the pulses in the training set have been determined with a positional accuracy of one period. A neural network trained for minimisation of an appropriately chosen AVR appears to operate correctly in this case, while a network trained with 'standard' error backpropagation fails.

References

Barnard, E. (1992). Optimisation for training neural nets. *IEEE Trans. Neural Networks* 3 (2), 232–240.

Dahl, E.D. (1987). Accelerated learning using the generalized delta rule. *Proc. 1st IEEE Internat. Conf. on Neural Networks*, San Diego, CA, June 21–24, Vol. II, 523–530.

Devijver, P.A. (1973). Relationships between statistical risks and the least-mean-square-error design criterion in pattern recognition. *Proc. 1st Internat. Joint Conf. on Pattern Recognition*, Washington, DC, 139–148.

Devijver, P.A. and J. Kittler (1982). *Pattern Recognition: A Statistical Approach*. Prentice-Hall, New York, 22–26.

Herz, J., A. Krogh and R.G. Palmer (1991). *Introduction to the Theory of Neural Computing*. Addison-Wesley, Reading, MA.

van der Heijden, F. (1992). A Statistical Approach to Edge and Line Detection in Digital Images. Dissertation, University of Twente, Netherlands, ISBN 90-9004922-3.

Holt, M.J.J. (1992). Comparison of generalization in multilayer perceptrons with the log-likelihood and least-squares

cost functions, *Proc. 11th IAPR Internat. Conf. on Pattern Recognition*, The Hague, Netherlands, Aug. 30 – Sept. 3, Vol. II, 17–20.

Holt, M.J.J. and S. Semmani (1990). Convergence of backpropagation in neural networks using a loglikelihood cost function. *Electronic Lett.* 26 (23), 1964–1965.

Hush, D.R. and J.M. Salas (1988). Improving the learning rate of back-propagation with the gradient reuse algorithm. *Proc. IEEE Internat. Conf. on Neural Networks*, San Diego, CA, July 24–27, Vol. I, 441–447.

Jacobs, R.A. (1988). Increased rates of convergence through learning rate adaptation. *Neural Networks* 1, 232–240.

Koplowitz, J. and T.A. Brown (1981). On the relation of performance to editing in nearest neighbour rules. *Pattern Recognition* 13 (3), 251–255.

Kraaijveld, M.A. (1993). Small Sample Behavior of Multi-layer Feedforward Network Classifiers: Theoretical and Practical Aspects. Dissertation, ISBN 90-6275-934-3/CIP.

Kung, S.Y. and J.N. Hwang (1988). An algebraic projection analysis for optimal hidden units size and learning rates in backpropagation learning. *Proc. IEEE Internat. Conf. on Neural Networks*, San Diego, CA, July 24–27, Vol. I, 363–370.

Nedeljkovic, V. (1992). A novel neural networks training algorithm that minimizes the probability of classification error. *Proc. 11th IAPR Internat. Conf. on Pattern Recognition*, The Hague, Aug. 30 – Sept. 3, Vol. II, 13–16.

Nedeljkovic, V. (1993). A novel multilayer neural networks training algorithm that minimizes the probability of classification error, *IEEE Trans. Neural Networks* 4 (4), 650–659.

Parker, D.B. (1987). Optimal algorithms for adaptive networks: second order backpropagation, second order direct propagation, and second order hebbian learning. *Proc. 1st IEEE Internat. Conf. on Neural Networks*, San Diego, CA, June 21–24, Vol. II, 593–600.

Raudys, S.J. and A.K. Jain (1991). Small sample size problems in designing artificial neural networks. In: I.K. Sethi and A.K. Jain, Eds., *Artificial Neural Networks and Statistical Pattern Recognition: Old and New Connections*. North-Holland, Amsterdam, 33–50.

Rumelhart, D.E., G.E. Hinton and R.J. Williams (1986). Learning internal representations by error propagation. In: D.E. Rumelhart and J. McClelland, Eds., *Parallel Distributed Processing, Explorations and the Microstructures of Cognition*, MIT Press, Cambridge, MA, 282–317.

Spreeuwers, L.J. (1991). A neural network edge detector. *Proc. SPIE/SPSE Symposium on Electrical Imaging Science & Technology, Nonlinear Image Processing II*, San Jose, CA, Feb. 24 – March 1. *SPIE* 1451, 205–215.

Spreeuwers, L.J. (1992). Image Filtering with Neural Networks. Applications and Performance Evaluation. Dissertation, University of Twente, Netherlands, ISBN 90-9005555-X.

Spreeuwers, L.J. and F. van der Heijden (1992a). An edge detector evaluation method based on average risk. *Proc. 2nd Internat. Workshop on Robust Computer Vision*, Bonn, Germany, March 9–12, 79–89.

Spreeuwers, L.J. and F. van der Heijden (1992b). Evaluation of edge detectors using average risk. *Proc. 11th IAPR Internat. Conf. on Pattern Recognition*, The Hague, Netherlands, Aug. 30 – Sept. 3, Vol. III, 771–774.

Spreeuwers, L.J. (1995). Optimisation for the sum of squared errors leads to estimation of the a posteriori conditional class probability density functions. Internal Report nr. 009M95, Laboratory for Measurement and Instrumentation, Department of Electrical Engineering University of Twente, Netherlands.