

PATTERN RECOGNITION

This invention relates to pattern recognition apparatus or the like using neural networks, and a method of producing such apparatus; in particular, but not exclusively, using neural networks of the multi-layer perceptron (MLP) type.

Neural networks of this type in general comprise a plurality of parallel processing units ("neurons"), each connected to receive a plurality of inputs comprising an input vector, each input being connected to one of a respective plurality of weighting units, the weighting factors of which comprise a respective weight vector, the output of the neuron being a scalar function of the input vector and the weight vector. Typically, the output is a function of the sum of the weighted inputs. Such networks have been proposed since the 1950s (or before), and have been applied to a wide variety of problems such as visual object recognition, speech recognition and text-to-speech conversion.

It is also known to implement such networks using a single, suitably programmed, digital computing device to perform the processing of all such neurons. Although the speed achievable by such an arrangement is of necessity of lower than that of a parallel network, the advantages of adaptive pattern recognition can give such an implementation greater speed and simplicity than would the use of conventional pattern recognition techniques.

The perceptron illustrated in Figure 1, consists of simple processing units (neurons) arranged in layers connected together via 'weights' (synapses). The output of each unit in a layer is the weighted sum of the outputs

from the previous layer. During training, the values of these weights are adjusted so that a pattern on the output layer is 'recognised' by a particular set of output units being activated above a threshold.

5 Interest in perceptrons faded in the 1960s, and did not revive again until the mid 1980s, when two innovations gave perceptrons new potential. The first was the provision of a non-linear compression following each neuron, which had the effect that the transformation
10 between layers was also be non-linear. This meant that, in theory at least, such a device was capable of performing complex, non-linear mappings. The second innovation was the invention of a weight-adjustment algorithm known as the 'generalised delta rule'. These innovations are discussed
15 in

Rumelhart, D.E., Hinton, G.E. & Williams, R.J. (1986). "Learning internal representations by error propagation."

In Parallel Distributed Processing

20 Eds McClelland & Rumelhart. MIT Press.

Since a sequence of non-linear transformations is not, generally, equivalent to a single, non-linear transformation, the new perceptron could have as many layers as necessary to perform its complicated mappings.
25 Thus the new device came to be known as the multi-layer perceptron (MLP). The generalised delta rule enabled it to learn patterns by a simple error back propagation training process. A pattern to be learned is supplied and latched ('clamped') to the input units of the device, and the
30 corresponding required output is presented to the output units. The weights, which connect input to output via the multiple layers, are adjusted so that the error between the actual and required output is reduced. The standard back propagation training algorithm for these networks employs a
35 gradient descent algorithm with weights and biases adjusted by an amount proportional to the gradient of the error function with respect to each weight. The constant of

proportionality is known as the learning rate. A 'momentum' term is also usually added that smooths successive weight updates by adding a constant proportion of the previous weight update to the current one. For large MLPs training on large amounts of data, an alternative algorithm computes a variable learning rate and momentum smoothing. The adaptation scheme ensures that steep gradients do not cause excessively large steps in weight space, but still permits reasonable step sizes with small gradients. This process is repeated many times for all the patterns in the training set. After an appropriate number of iterations, the MLP will recognise the patterns in the training set. If the data is structured, and if the training set is representative, then the MLP will also recognise patterns not in the training set.

Such training techniques are discussed in, for example, British Telecom Technology Journal Vol 6, No. 2 April 1988 p131-139 "Multi-layer perceptions applied to speech technology", N. McCulloch et al, and are well known in the art. Similar training procedures are used for other types of neural network.

To be effective, the network must learn an underlying mapping from input patterns to output patterns by using a sparse set of examples (training data). This mapping should also be applicable to previously unseen data, i.e. the network should generalise well. This is especially important for pattern classification systems in which the data forms natural clusters in the input feature space, such as speech data.

In the following, generalisation is defined as the difference between the classification performance on the training data set and that on a test set drawn from a population with the same underlying statistics. Why does a given network fail to generalise? A net is specified by a set of parameters that must be learned from a set of training examples. If the amount of training data available is increased, the better in general will be the weight estimates and the more likely a net will

generalise. In all practical applications, the amount of training data available is limited and strategies must be developed to constrain a network so that a limited amount of data will produce good weight values. Limiting the internal complexity of the network (numbers of hidden nodes and connectivity) is one prior method of constraint.

The standard algorithm used for training multi-layer perceptrons is the error back-propagation algorithm discussed above.

To learn a particular mapping, the algorithm adjusts network weight and bias values so as to reduce the sum-squared error between the actual network output and some desired output value. In a classification task such as speech or visual pattern recognition, the correct output is a vector that represents a particular class. However, the same class label will be assigned to an input vector whether or not it is close to a class boundary. Boundary effects, and as importantly for "real-world" problems, noise at the level of the feature description, mean that, to minimise the output error, fast transitions of the output nodes in the input space are required. To build up fast transitions, large weight values need to be used. The back-propagation algorithm, if run for a large number of iterations, builds up such large values.

It can be seen that generalisation in the presence of noise and limited training data is promoted if smooth decision surfaces are formed in the input feature space. This means that a small change in input values will lead to only a relatively small change in output value. This smoothness can be guaranteed if the connection weight magnitudes are kept to low values. Although it may not be possible for the network to learn the training set to such a high degree of accuracy, the difference between training set and test set performance decreases and test set performance can increase.

Previously, the generalisation problem has been tackled simply by initialising a network using small random weights and stopping the training process after a

small number of cycles. This is done so that the training set should not be learnt in too great detail. Additionally, some workers (for example Haffner et al "Fast Back-propagation learning methods for phonemic neural networks". Eurospeech 89), Paris have realised that large network weight values lead to poor generalisation ability, and that it is for this reason that training cycles should be limited. However, the correct number of training cycles to choose is dependent on the problem, the network size, network connectivity and on the learning algorithm parameters. Hence, simply limiting training cycles is a poor solution to an important problem.

In any digital hardware implementation of an MLP or other network, the question of weight quantisation must be addressed. It is known that biological neurons do not perform precise arithmetic, so it might be hoped that weights in a neural network would be robust to quantisation. Normally, quantisation takes place after the network has been trained. However, if as described above, the network has built up large weight values, then node output may depend on small differences between large values. This is an undesirable situation in any numerical computation, especially one in which robustness to quantisation errors is required.

The issue of weight-quantisation is also an area that has not been approached with a view to performing MLP training subject to a criteria that will improve quantisation performance. MLP weights would normally be examined after training and then a suitable quantisation scheme devised. It is true that the prior art technique of limiting the number of training cycles as discussed above, will improve weight quantisation simply because weight values will have not yet grown to large values, but the extent to which it does so depends, as discussed above, on a number of parameters which may be data-related. It is thus not a general-purpose solution to the problem of providing readily quantised weights.

We have found that both generalisation performance and robustness to weight quantisation are improved by including explicit weight-range limiting into the MLP training procedure.

5 According to the invention there is provided:

- a method of deriving weight vectors (each comprising a plurality of multibit digital weights each having one of a range of discrete values) for a neural net comprising vector input means for receiving a plurality of
10 input values comprising an input vector; and vector processing means for generating a plurality of scalar outputs in dependence upon the input vector and respective reference weight vectors,

comprising the steps of: -

15 - selecting a sequence of sample input vectors (corresponding to predetermined net outputs);

- generating therefrom an approximation to the outputs of the net corresponding to the respective input vectors; and

20 - iteratively modifying the weight vectors so as to reduce the difference between the said approximated net outputs and the predetermined outputs; and when this modifying step results in the magnitude of a weight, or weight vector, exceeding a predetermined value M, then that
25 weight, or weight vector, magnitude is constrained to be equal to (or less than) M; and

- subsequently quantising the thus derived weights or weight vectors.

30 Our results clearly show the effectiveness of the weight range-limiting technique in both improving quantisation performance and increasing the robustness of the structure to weight quantisation, although it may seem surprising that good test-set accuracy can be obtained by

networks with very limited weight ranges (down to ± 1.5). It is important to note that good performance is due to the fact that the weight-limiting technique is incorporated into the learning procedure and hence the MLP parameters are optimised subject to constraints on the weight values. This process can be thought of as incorporating knowledge into the structure by disallowing weight-space configuration that will give poor generalisation due to the inherent noise in data for real-world problems.

Weight limiting also improves general network robustness to numerical inaccuracies - hence weight quantisation performance improves. It is seen here that with suitable weight limits as few as three bits per weight can be used. Large weights can cause a network node to compute small differences between large (weighted) inputs. This in turn gives rise to sensitivity to numerical inaccuracies and leads to a lessening of the inherent robustness of the MLP structure. The weight limited MLP according to the invention is able to deal with inaccuracies in activation function evaluation and low resolution arithmetic better than a network with larger weight values. These factors combine to mean that the technique is useful in any limited precision, fixed-point MLP implementation. The numerical robustness is increased so that, with suitable weight limits, as few as three bits per weight can be used to represent trained weight values.

Simulations on a "real-world" speech recognition problem show that, for a fixed MLP structure, although classification performance on the training set decreases, improved generalisation leads to improved test set performance.

A neural network employing weights derived according to the above method will be distinguishable, in general, from prior art networks because the distribution of the weight values will not be even (i.e. tailing off at high positive and negative weight sizes), but will be skewed

towards the maximum level M , with a substantial proportion of weight magnitudes equal to M . It will also, as discussed above, have an improved generalisation performance. The invention thus extends to such a network.

5 Such a network (which need not be a multi-layer perceptron network but could be, for example, a single layer perception), is, as discussed above, useful for speech recognition, but is also useful in visual object recognition and other classification tasks, and in
10 estimation tasks such as echo cancelling or optimisation tasks such as telephone network management. In some applications (for example, speaker-dependent recognition), it may be useful to supply an "untrained" network which includes training means (e.g. a microcomputer) programmed
15 to accept a series of test data input vectors and derive the weights therefrom in the manner discussed above during a training phase.

20 The magnitude referred to is generally the absolute (i.e. $\pm$) magnitude, and preferably the magnitude of any single weight (i.e. vector component) is constrained not to exceed M .

The invention will now be described by way of example only, with reference to the accompanying drawings, in which:-

25 - Figure 1 shows schematically a (prior art) perceptron;

- Figure 2 shows schematically a multi-layer perceptron;

30 - Figure 3 shows schematically a training algorithm according to one aspect of the invention

- Figure 4a shows schematically an input stage for an embodiment of the invention,

- Figure 4b shows schematically a neuron for an embodiment of the invention, and

35 - Figure 5 shows schematically a network of such neurons.

The following describes an experimental evaluation of the weight range-limiting method of the invention. First, the example problem and associated database are described, followed by a series of experiments and the results.

5 The problem was required to be a "real-world" problem having noisy features but also being of limited dimension.

 The problem chosen was the one of speaker independent recognition of telephone quality utterances of "yes" and "no" using a simple feature description. This task has
10 all the required properties listed above and also the added advantages of known performance using several other techniques; see

 Woodland, P.C. & Millar, W. (1989). "Fixed dimensional classifiers for speech recognition". In
15 Speech and Language Processing Eds. Wheddon & Linggard. Kogan-Page.

 The speech data used was part of a large database collected by British Telecom from long distance telephone talks over the public switched telephone network. It
20 consisted of a single utterance of the words "yes" and "no" from more than 700 talkers. 798 utterances were available for MLP training and a further 620 for testing. The talkers in the training set were completely distinct from the test set talkers. The speech was digitally
25 sampled at 8kHz and manually endpointed. The resulting data set included samples with impulsive switching noise and very high background noise levels.

 The data was processed by an energy based segmentation scheme into five variable length portions. Within each
30 segment low order LPC analysis was used to produce two cepstral coefficients and the normalised prediction error. The complete utterance was therefore described by a single fifteen dimensional vector.

 A number of MLPs were trained using this database to
35 assess the effects of weight range-limiting on generalisation and weight quantisation. In all cases the MLPs had a single hidden layer, full connection between the input and hidden layers and full connection between

the hidden and output layer. There were no direct input/output connections. The back-propagation training algorithm was used with updating after each epoch, i.e., after every input/output pattern has been presented. This update scheme ensures that the weights are changed in a direction that reduces error over all the training patterns and result of the training procedure does not depend on the order of pattern presentation. All networks used a single output node. During training, the desired output was set to 0.9 for "yes" and to 0.1 for "no". During testing all utterances that gave an output of greater than 0.5 were classified as "yes" and the remainder as "no". The learning rate used was 0.01 and the momentum constant was set to 0.9.

Weight range-limited networks were trained subject to a maximum absolute value of M . If after a weight update a particular weight exceeded this value then it was reset to the maximum value. Classification performance results for values of M of 1.5, 2.0, 3.0, 5.0 and no-limit are shown as Tables 1 to 5

TABLE 1 - Performance with $M = 1.5$

Number of Hidden Nodes	Training Set Accuracy%	Test Set Accuracy%
2	90.4	90.7
3	92.2	92.3
4	92.7	93.2
5	93.9	94.7
10	95.7	95.5
15	96.6	95.7
20	97.1	95.5
25	97.4	95.3
30	97.0	95.5

TABLE 2 - Performance with M = 2.0

Number of Hidden Nodes	Training Set Accuracy%	Test Set Accuracy%
2	92.9	92.7
3	94.1	94.0
4	94.6	95.2
5	94.9	95.3
10	96.9	95.0
15	97.9	95.7
20	97.6	95.3
25	98.1	96.3
30	97.9	95.2

TABLE 3 - Performance with M = 3.0

Number Of Hidden Nodes	Training Set Accuracy %	Test Set Accuracy %
2	94.4	95.0
3	95.2	95.2
4	96.2	95.3
5	97.0	94.8
10	98.9	95.3
15	99.0	95.2

TABLE 4 - Performance with $M = 5.0$

Number of Hidden Nodes	Training Set Accuracy %	Test Set Accuracy %
2	95.6	95.5
3	96.5	95.3
4	97.6	94.7
5	97.9	95.3
10	99.5	94.3
15	99.5	93.7

TABLE 5 - Performance with no weight limits

Number of Hidden Nodes	Training Set Accuracy %	Test Set Accuracy %
2	97.0	95.2
3	97.9	93.7
4	98.6	95.0
5	98.9	94.5
10	99.8	94.0
15	99.8	94.0

5

It can be seen from these tables that as M is increased the training set accuracy increases for a given number of hidden nodes, tending towards the figures with no weight constraints. It can also be seen that training set and test set accuracies converge as M is decreased, i.e., generalisation performance improves. Further, as the number of hidden nodes increase, the test set performance gets worse for the no limit case, while

(M=) for small values of M this effect is not in evidence. It should be noted that the best test set performance occurs for $M = 2.0$ with 25 hidden nodes. Also, for all finite values of M tested, there is at least one MLP configuration that gives superior test-set performance to the best achieved with no weight limiting.

Experiments to ascertain the relative degradations due to weight quantisation were also performed. Once the network had been fully trained using a floating point arithmetic training algorithm, the weights and biases were quantised into a fixed (odd) number, L, of equally spaced levels. The maximum absolute value in the network determined the maximum positive and maximum negative level value. Each network weight and bias was then mapped to the nearest quantisation level value and network testing performed using floating point arithmetic. The effects of weight quantisation after training was tested for the 15 hidden node networks described above. Table 6 gives the RMS error (between actual and desired outputs) on the training set for different numbers of levels and a range of maximum absolute weights.

TABLE 6 - RMS Error on training set for differing L & M

No of Levels L	RMS Error on Training Set				
	M = 1.5	M = 2.0	M = 3.0	M = 5.0	M = ∞
3	0.2674	0.1700	0.1954	0.4849	0.5391
7	0.1458	0.1540	0.1374	0.4514	0.4993
15	0.1465	0.1277	0.1162	0.1920	0.3770
31	0.1445	0.1255	0.1212	0.1000	0.1428
63	0.1443	0.1254	0.1079	0.0975	0.0977
127	0.1441	0.1253	0.1079	0.0970	0.1205
255	0.1441	0.1253	0.1080	0.0962	0.0958
∞	0.1441	0.1253	0.1077	0.0961	0.0923

From Table 6 the approximate number of levels required, and hence the number of bits per weight, for no

significant degradation can be calculated. Table 7 lists the number of bits per weight required such that the RMS error is less than 5% greater than for unquantised weights for each of the values of M under consideration.

5 TABLE 7 - Bits/weight required
Maximum weight Bits per weight
M required

1.5	3
2.0	4
3.0	6
5.0	5
∞	8

Although, as discussed, the invention is primarily intended for implementation in neural networks of the well known perceptron type which need no further description to the skilled man, some exemplary embodiments will now be briefly disclosed with reference to Figures 4a, 4b & 5 (with particular reference to speech recognition).

10 A real-world input (speech) signal is received, sampled, digitised, and processed to extract a feature vector by an input processing circuit 1. For a speech input, the sampling rate may be 8 KHZ; the digitisation may involve A-law PCM coding; and the feature vector may for example comprise a set of LPC coefficients, Fourier Transform coefficients or preferably melfrequency cepstral coefficients.

20 For speech, the start and end points of the utterance are determined by end pointing device 1a, using for example the method described in Wilpon J.G, Rabiner and Martin T: 'An improved word-detection algorithm for telephone quality speech incorporating both syntactic and semantic constraints', AT&T Bell Labs Tech J, 63 (1984) (or any other well known method), and between these points n-dimensional feature vectors X are supplied periodically (for example, every 10-100 msec) to the input node 2 of

the net. The vector X may be a set of 15 8-bit coefficients x_i (i.e. $n=15$)

5 The feature vectors are conveniently supplied in time-division multiplexed form to a single input 2a connected to common data bus 2b.

The input bus is connectable to each neuron 3a, 3b, 3d, in the input layer.

Referring to Figure 4b, each comprises a weight vector ROM 5 storing the 15 weight values. The clock
10 signal controlling the multiplex also controls the address bus of the ROM 5, so that as successive input values x_i ($i = 1$ to n) of the input vector X are successively supplied to a first input of hardware digital multiplier 6, corresponding weight values w_i are
15 addressed and placed on the data bus of the ROM 5 and supplied to a second input of the multiplier 6. The weighted input value produced at the output of the multiplier 6 is supplied to a digital accumulating adder 7, and added to the current accumulated total.

20 The clock signal also controls a latch 8 on the output of the accumulating adder 7 so that when all weighted input values of an input vector are accumulated, the (scalar) total is latched in latch 8 for the duration of the next cycle. This is achieved by dividing the clock
25 by n . The accumulating adder 7 is then reset to zero (or to any desired predetermined bias value). The total $y (= \sum x_i \cdot w_i)$ is supplied to a non linear compression circuit 9 which generates in response an output of compressed range, typically given by the function

$$C(y) = 1/(1+\exp(-y))$$

30 The compression circuit 9 could be a lookup ROM, the total being supplied to the address lines and the output being supplied from the data lines but is preferably as disclosed in our UK application GB8922528.8 and the article 'Efficient Implementation of piecewise
35 linear activation function for digital VLSI Neural Networks'; Myers et al, Electronics letters, 23 November 1989, Vol 25 no 24.

Referring to Figure 5, the (latched) scalar output Y of each neuron is connected as an input to the neurons 10a, 10b, 10c of the intermediate or 'hidden' layer. Typically, fewer neurons will be required in the hidden layer. The output values may be multiplexed onto a common intermediate bus 11 (but clocked at a rate of 1/n that of the input values by, for example, a flipflop circuit) in which case each hidden layer neuron 10a, 10b, 10c may be equivalent to the input layer neurons 3a, 3b, 3c, receiving as their input vector the output values of the input layer. This is a useful embodiment in that it allows a flexible implementation of various nets with low interconnection complexity.

The neurons 10a, 10b, 10c of the hidden layer likewise supply scalar outputs which act as inputs to neurons 12a, 12b, of the output layer. In a classification application, there is typically an output neuron for each class (and possibly others; for example one for "not recognised"). For a speech recogniser intended to recognise "yes" and "no" two output neurons 12a, 12b could be provided as shown (although a single output could be used to discriminate between two classes). Each receives as its input vector the set of outputs of the hidden layer below, and applies its weight vector (stored, as above, in ROMs) via a multiplier, to produce a net output value. Preferably the output layer neurons also use a compression function. The class which corresponds to the output layer neuron producing the largest output value is allocated to the input vector (although other 'decision rules' could be used).

Various modifications and substitutions to the above example are possible.

Further intermediate layers may be included. The skilled man will realise that the functions of all neurons of one input layer(s) could be performed by a single computing device sequentially, in environments where high speed is inessential; the multiply and accumulate/add operations are of course common in microprocessors and in dedicated digital signal processing

(DSP) devices such as the Texas Instruments[™] TMS320 family or the Motorola[™] TM56000 family.

Equally, all the ROMs 5 could be realised as areas of a single memory device. Alternatively, the invention may
5 be realised using hybrid analogue digital networks in which digital weights act on analogue signals, using neurons of the type shown in Figs 11 and 12 of WO 89/02134 (assigned to the present applicants).

In applications such as hybrid video coding, speaker
10 recognition or speaker-dependent speech recognition, the network needs to be 'trainable' - that is, must be capable of devising satisfactory weight values on training data. Accordingly, in this type of apparatus a weight adjusting device 13 is provided; typically a microprocessor operating
15 according to a stored program. The input test patterns are supplied to the input 1, and the initial weight values are adjusted, using an error back-propagation algorithm, to reduce the difference between the net outputs (to which the device 13 is connectable) and the predetermined outputs
20 corresponding to the inputs. The device 13 thus calculates the output error, calculates the necessary weight value increment for each weight; limits the weight magnitudes (if necessary) to M; accesses the weights in each store 5 (which must, of course in this embodiment, be a read/write
25 store, not a ROM), adds the increments and rewrites the new weight values to the stores 5; the method is discussed above. Consequently, the adjusting device 13 is connected to the address busses of all stores 5 and to the net outputs. It is also connected to a source of correct
30 output values; for example, in training a speaker-dependant word recogniser these are derived from a prompt device (not shown) which instructs the speaker to speak a given word. The correct output (say 0.9) for the output neuron corresponding to that word, is supplied to weight adjuster
35 13 together with correct outputs (say 0.1) for all other output neurons.



CLAIMS

1. A method of deriving weight vectors (each comprising a plurality of multibit digital weights each having one of a range of discrete values) for a neural net comprising:-
 - vector input means for receiving a plurality of
 - 5 input values comprising an input vector; and
 - vector processing means for generating a plurality of scalar outputs in dependence upon the input vector and respective reference weight vectors, comprising the steps of:-
 - 10 - selecting a sequence of sample input vectors which correspond to predetermined net outputs;
 - generating therefrom an approximation to the outputs of the net corresponding to the respective input vectors; and
 - 15 - iteratively modifying the weight vectors so as to reduce the difference between the said approximated net outputs and the predetermined net outputs and, when this modifying step results in the magnitude of a weight, or weight vector, exceeding a predetermined value M, then that
 - 20 weight, or weight vector, magnitude is constrained to be equal to (or less than) M; and
 - subsequently quantising the thus modified weights or weight vectors.
2. A method according to claim 1, further comprising
- 25 quantising the thus-modified weights or said weight vectors to a lower precision, one of the quantised levels being said value M.
3. A trainable neural network comprising:-
 - vector input means for receiving an input vector;
 - 30 and
 - vector processing means for generating a plurality of scalar outputs in dependence upon the input vector and a plurality of respective reference, weight,



vectors each comprising a plurality of multibit digital weights each having one of a range of discrete values, further comprising: -

5 - training means for deriving, in a training phase, the weights of said reference weight vectors, the training means including means for limiting, during the training phase, said reference weight vectors or weights to a predetermined maximum value M by constraining the value of weight vectors or weights which would otherwise exceed M
10 not to so do; and quantising means for quantising said thus-derived weights subsequent to said training phase.

4. A network according to claim 3 arranged to comprise a multi-layer perceptron.

15 5. A neural network having weight vectors derived by the method of claim 1 or claim 2.

6. A neural network according to claim 5, connected to comprise a multi-layer perceptron.

20 7. A neural network according to any one of claims 3 to 7 in which the distribution of weight values is skewed towards a maximum magnitude equal to M, a substantial number of said weights having said maximum magnitude.

8. A method of training a neural network substantially as described herein, with reference to the accompanying Figure 3.

25 9. A neural network substantially as herein described with reference to Figures 3 to 5 of the accompanying drawings.
