Experiments with multi-layer perceptrons

J.L. Cybulski[1], H.L. Ferrá[2], A. Kowalczyk[3] and J. Szymański[4]

[1,3,4] *Artificial Intelligence Systems*
Telecom Research Laboratories
770 Blackburn Road
Clayton, Vic. 3168
Australia

[2] *Department of Mathematics*
Monash University
Clayton, Vic. 3168
Australia

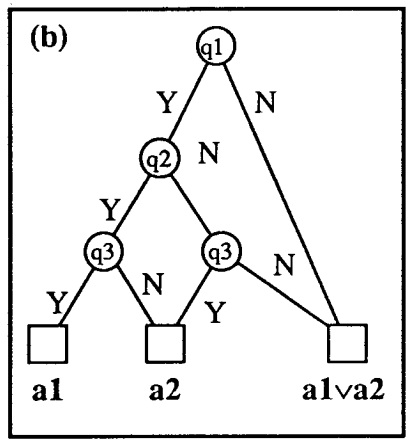
Abstract

This paper reviews results of experiments with three different classes of multi-layer perceptrons. The experiments performed ranged from simple deterministic and noisy pattern recognition, to the large size classification problems in natural language parsing. The experimental results reveal some of the benefits and disadvantages of different models, their space/time requirements, and finally their applicability to different problem domains.

1 Introduction. *Decision tables* (cf. Fig. 1.a) are often used as one of the most straightforward techniques for decision making. They appear implicitly or explicitly in numerous applications in different areas of Artificial Intelligence, e.g. in Vision, Speech Recognition, Natural Language Understanding and Expert Systems. In spite of their simplicity, decision tables are quite cumbersome to use in practice. They consist of a number of decision rules formed by explicit enumeration of all possible combinations of observable control attribute values and corresponding actions (decisions). The number of such rules often rises exponentially with the number of control attributes which clearly increases the computational complexity of the decision table processing. As an alternative solution, it is possible to generate (or calculate) decisions dynamically from the observed attribute values. *Decision trees* (Breiman, *et al.* 1984, Quinlan 1979, 1983; see also Fig 1.b) and *multi-layer perceptrons* (cf. Section 1.1 and Fig 1.c) are two examples of such dynamic decision systems. This paper focuses on various issues related to some classes of perceptrons and will occasionally refer to decision trees for comparison reason only.

1.1 Perceptron Structure. *Two layer perceptrons* (or just *perceptrons*) are collections of processors of limited capacity and storage interconnected by weighted links. The processors are repositories of values also known as activations and are arranged in two layers of active units, hidden and output, and an additional layer of passive input units. Perceptrons associate appropriate output patterns with given sets of input values by massively parallel computation and communication of activations (Minsky and Papert 1969, Rumelhart and McClelland 1986, Lippman 1987).

(a)	*I1*	*I2*	*I3*	*I4*	*I5*	*I6*	*I7*
q1	Y	Y	Y	N	Y	N	N
q2	Y	Y	N	Y	N	Y	N
q3	Y	N	Y	Y	N	N	Y
q4	Y	N	Y	Y	N	N	N
Prob.	0.25	0.10	0.10	0.10	0.05	0.10	0.20
a1	1	0	0	1	1	1	1
a2	0	1	1	1	1	1	1



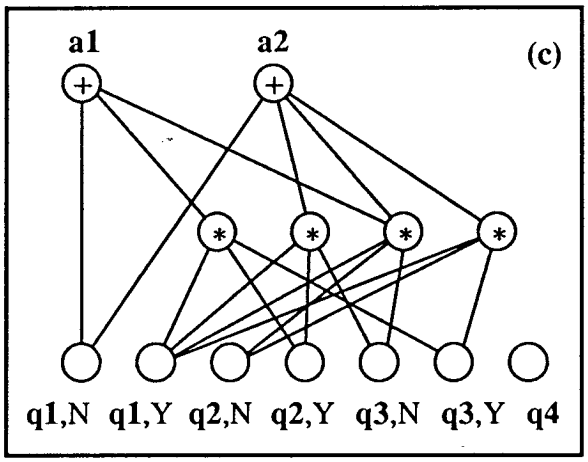


Fig. 1 - Illustration of three approaches to decision making: (a) decision table, (b) decision tree and (c) two layer (integer) perceptron

In this paper, we consider only a restricted class of perceptrons (Fig. 2). We assume all input units to have binary activation (i.e. 0 and 1 values), all hidden units to calculate logical conjunctions of subsets of inputs also resulting in their binary activations (cf. conjunction is also implementable with the 'threshold logic'), and finally output units to calculate the weighted sum of all lower layer unit activations (note that the zero weight connections are routinely omitted). Depending on the nature of these weights three different classes of perceptrons may be distinguished :-

(i) *Real perceptrons*. Weights α,β, ... are real numbers and the sum in the output is an ordinary arithmetic sum.

(ii) *Integer perceptrons*. They are real perceptrons with integer weights.

(iii) *Binary perceptrons*. In this case the weights $\alpha, \beta,... \in \{0, 1\}$ and the output sums are taken mod 2 (this corresponds to logical XOR).

In the real and integer cases the positive links are denoted by the solid and the negative ones by the broken lines.

Note that the binary perceptrons may be obtained by formally 'factorising by 2' all weights of

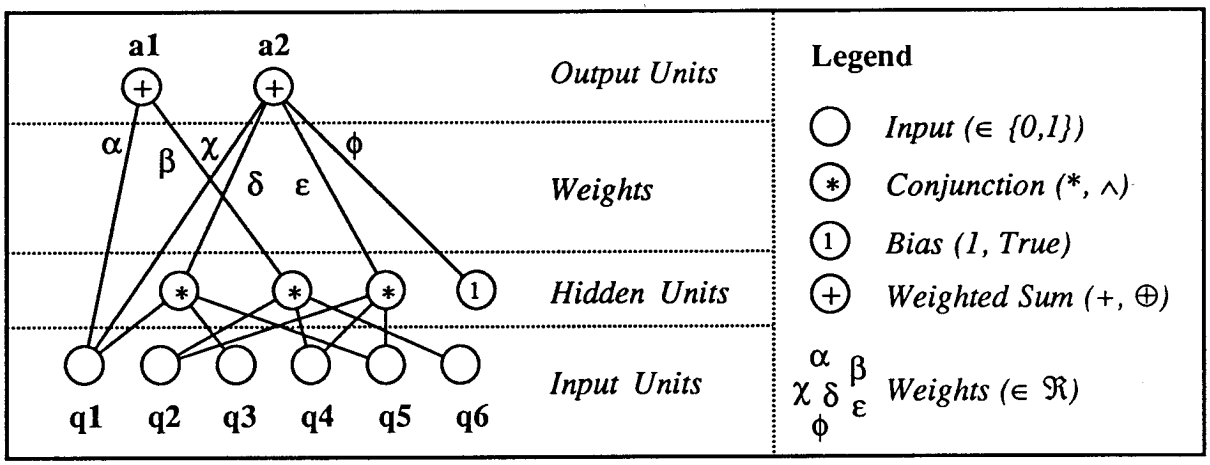


Fig. 2 - A two layer perceptron

integer perceptrons, i.e. by retaining all the odd weighted links and discarding all of the even ones.

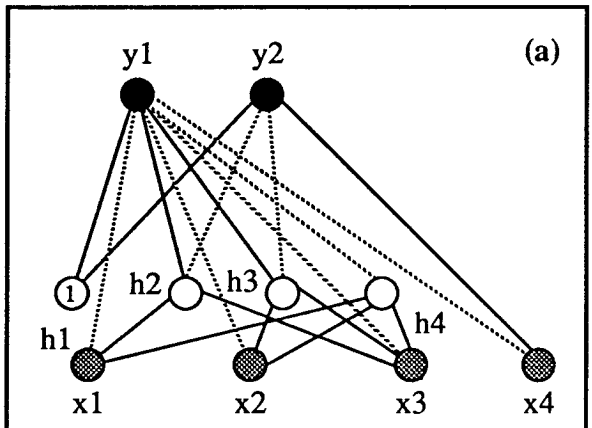
Our perceptrons differ from the typical 2-layer perceptrons (Minsky and Papert 1969, Lippman 1987) in respect to the following:

(i) the weights of links from input to hidden units are binary rather than real values,
(ii) our hidden units, performing logical conjunctions of their inputs (or equivalently multiplication), can only be viewed as a subclass of more general threshold units, and
(iii) the thresholds in output units are only optional.

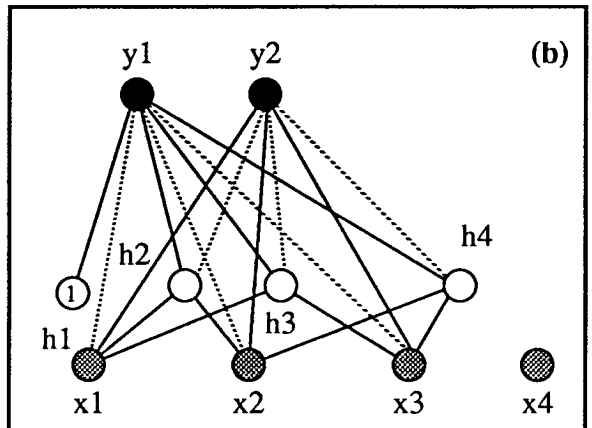
It is not difficult to see that any deterministic transformation from a finite set of finite bit strings into a set of the finite strings of real numbers (or strings of bits) can be implemented by a real (or a binary) perceptron of the above form. Thus, the structure is quite general. The thresholds in the outputs may be added as necessary. However, the explicit consideration of linear outputs in early stages of training allows more freedom (e.g. by taking the largest of outputs, if appropriate - cf. Section 3.2) and simplify our training algorithms. Moreover, from Widrow and Hoff (1960) it follows that if such a 'linear perceptron' is first trained in terms of minimizing its mean-square-error, then an imposition of a threshold on each of its outputs does not significantly affect the perceptron's overall performance.

As an illustration, let us consider a simple decision table in Fig. 1.a containing a set of seven rules for four binary control attributes *q1*,...,*q4* ('Y' and 'N' values) and two binary decisions *a1* and *a2* ('0' and '1' values). A decision tree in Fig. 1.b and a two-layer integer perceptron in Fig. 1.c are equivalent to this decision table. It is also worthwhile mentioning that typically, for a given problem, there is a number of different perceptrons of equivalent performance, for instance, Fig. 3 depicts five different perceptron architectures of performance identical to that of the perceptron in Fig. 1.c. (note that *xi* in Fig 3 is equivalent to *qi,Y* in Fig 1.c).

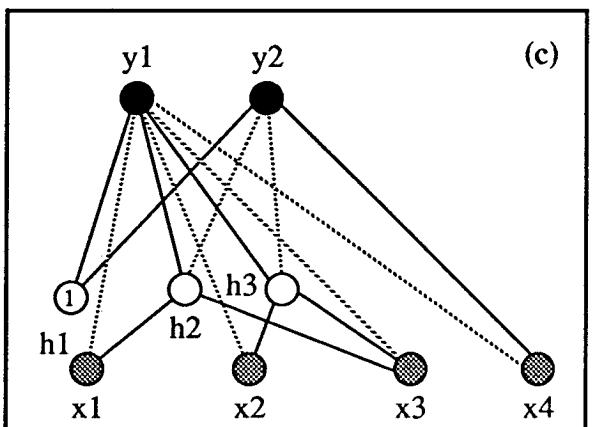
1.2 Remarks on training algorithm. A perceptron applicable to a given task (i.e. performing a specific set of input/output transformations) may be constructed either algorithmically or by applying an appropriate training algorithm to the a set of selected exemplars (Minsky and Papert 1969, Grossberg 1980, Hopfield 1982, Ackley, Hinton and Sejnowski 1985).



x1 y1 -1.0	h1 y1 +2.0	h3 y2 -1.0
x2 y1 -1.0	h1 y2 +1.0	h4 y1 -0.5
x3 y1 -1.0	h2 y1 +1.5	
x4 y1 -1.5	h2 y2 -1.0	
x4 y2 +1.0	h3 y1 +2.5	



x1 y1 -2	x3 y2 +1	h3 y2 -1
x1 y2 +1	h1 y1 +3	h4 y1 +2
x2 y1 -2	h2 y1 +1	h4 y2 -1
x2 y2 +1	h2 y2 -1	
x3 y1 -2	h3 y1 +1	



x1 y1 -1	h1 y1 +2	h3 y2 -1
x2 y1 -1	h1 y2 +1	
x3 y1 -1	h2 y1 +1	
x4 y1 -1	h2 y2 -1	
x4 y2 +1	h3 y1 +2	

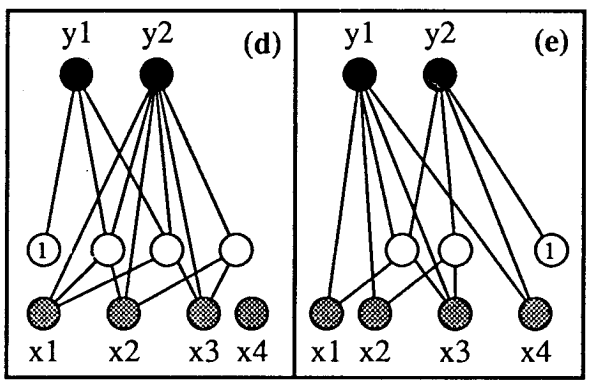


Fig. 3. Five different perceptrons implementing decision table in Fig 1.a:

(a) a real perceptron,
(b, c) two integer perceptrons, and
(d , e) two binary perceptrons

Note that weights for non-binary perceptrons are given in a tabular form

Back-propagation (Rumelhart, Hinton and Williams 1986) is the most popular algorithm for the design of perceptrons. It starts with a totally interconnected perceptron structure with random weights. Then the weights are gradually changed in the direction of minimization of m.s.e. between the actual and desired outputs for each of the exemplar inputs. The procedure requires typically a considerable number of passes of data, manipulation of different control parameters, and significant computational power. Additionally, the method demands active units to have differentiable non-linearities (usually sigmoid), which forces large weights, unsatisfactory ratio between the largest and the smallest of weights, etc. (Cf.

Epilogue of Minsky and Papert, 1988 edition.) Nevertheless, the method was proven to produce satisfactory results in a number of applications.

It is reasonable to expect that a back-propagation approach could be considerably enhanced if started with initial 'roughly correct' perceptron structure rather than a random one. Hence, an alternative approach to the perceptron design, as outlined below, could be utilized for such an initialization.

Our algorithms perform a heuristic-controlled search for, and addition of, suitable hidden units, generally from their lower to the higher order (Cybulski, Kowalczyk, Szymanski 1989, Kowalczyk 1989a,b). The crux of the approach is in a relatively quick algebra-based heuristic assessment of usefulness of the proposed new addition to the list of already selected units. This saves a lot of computer time, since it is not required to continuously recalculate the

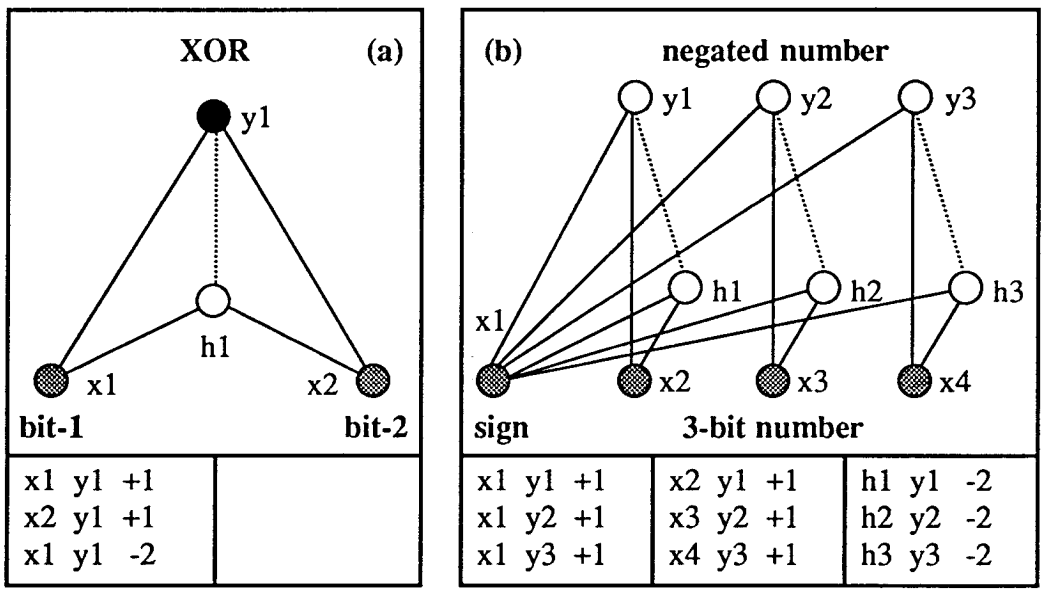


Fig. 4. Integer perceptrons generated for (a) XOR and (b) negation problem.

weights as in a back-propagation approach. Actually, the weights are calculated once only after the perceptron structure is determined. The fine tuning by retraining methods, as in back-propagation, or weight rounding may be used to further improve performance or to simplify the perceptron structure as desired.

As any other algorithm, our training algorithm has its own drawbacks and limitations. Our initial research aimed at exploring some of them. It has been a nice surprise to us, that those boundaries turned out to be further than we anticipated.

The algorithm is memory extensive and is very vulnerable to the number of potential inputs to be considered. Experiments indicate that the CPU time increases proportionally with the size of the training set raised to the power equal to the highest order of the hidden units needed. Any preprocessing selecting the subset of essential inputs (cf. entropy approach

Kowalczyk 1989a, rough set approach - Pawlak, Wong and Ziarko 1987) is therefore helpful in reducing this complexity.

In this paper we intended to illustrate by examples some of the achievements and limitations of our approach. In particular we list the CPU time used to generate the examples considered in this paper. This should give some idea of the practical complexity of the approach. To this purpose, we started with two simple experiments (XOR and negation) which were earlier reported as tests of the back-propagation algorithm (Rumelhart, Hinton and Williams 1986). To generate the perceptrons in Fig. 4 it took 0.1s and 0.64s of CPU on SUN 3/50. The back propagation solutions had exactly the same architecture, except that the weights were not so uniform and they were ranging between -10 and 6.3, and the hidden and output units provided the sums with superimposed sigmoid non-linearity. It was reported that it took ~450 and 5000 passes of data to obtain the solutions for XOR and negation, respectively. However, these results do not give a fair comparison with back-propagation, as Rumelhart, Hinton and Williams 1986 gives only the number of passes through the data rather than the CPU time.

2 Deterministic classification by perceptrons with real coefficients

The aim of the exercise was to test and refine the training algorithm for building real perceptrons in a relatively simple setting. A set of 20 upper case characters stylised on a 5×5 on-off grid was used as a basis for this series of experiments (Fig. 5). All perceptrons generated here had 25 inputs, corresponding to the units of the grid, and 20 output units corresponding to the characters A, B,...,T (cf. Fig 5.a). It was intended that for each valid grid pattern the corresponding output unit would be set to 1 (the character identification) with all of the remaining outputs being 0.

2.1 Plain character recognition. In this experiment we constructed a perceptron to recognize characters in Fig 5.a. We found a perceptron with two hidden units of the second order was sufficient for this task. It took about 4s. of CPU to generate it on the SUN 3/50. All weights obtained were rational, with denominator 2, and values between -3 and 2.

2.2 Recognition of rotated characters. In this experiment each character class had four exemplars representing the rotated versions of each of the 20 characters (cf. Fig. 5.b) so altogether we had 80 exemplars divided into 20 classes. We constructed a 100% accurate perceptron with ~400 connections and 44 hidden units. Its generation time on SUN 3/50 was 40s.

2.3 Recognition of periodically shifted characters. In this case each of the 20 characters (cf. Fig. 5.a) was periodically shifted 25 times (cf. Fig. 5.c) giving 500 exemplars. The experiment was inspired by a simple idea of a position-independent character recogniser with a 25 units toroidal concentrator (illustrated in Fig. 5.d) which, when unfolded, could be mapped onto a 5×5 grid. Each of the units of the infinite plane is (periodically) connected to one of the units of the toroidal concentrator, e.g. the unit *(i, j)* of the plane is connected to the unit *(i mod 5, j mod 5)* of the concentrator. Thus, each of the characters, in any of the (non-rotated!) positions in the plane, leads to one of the periodically shifted patterns.

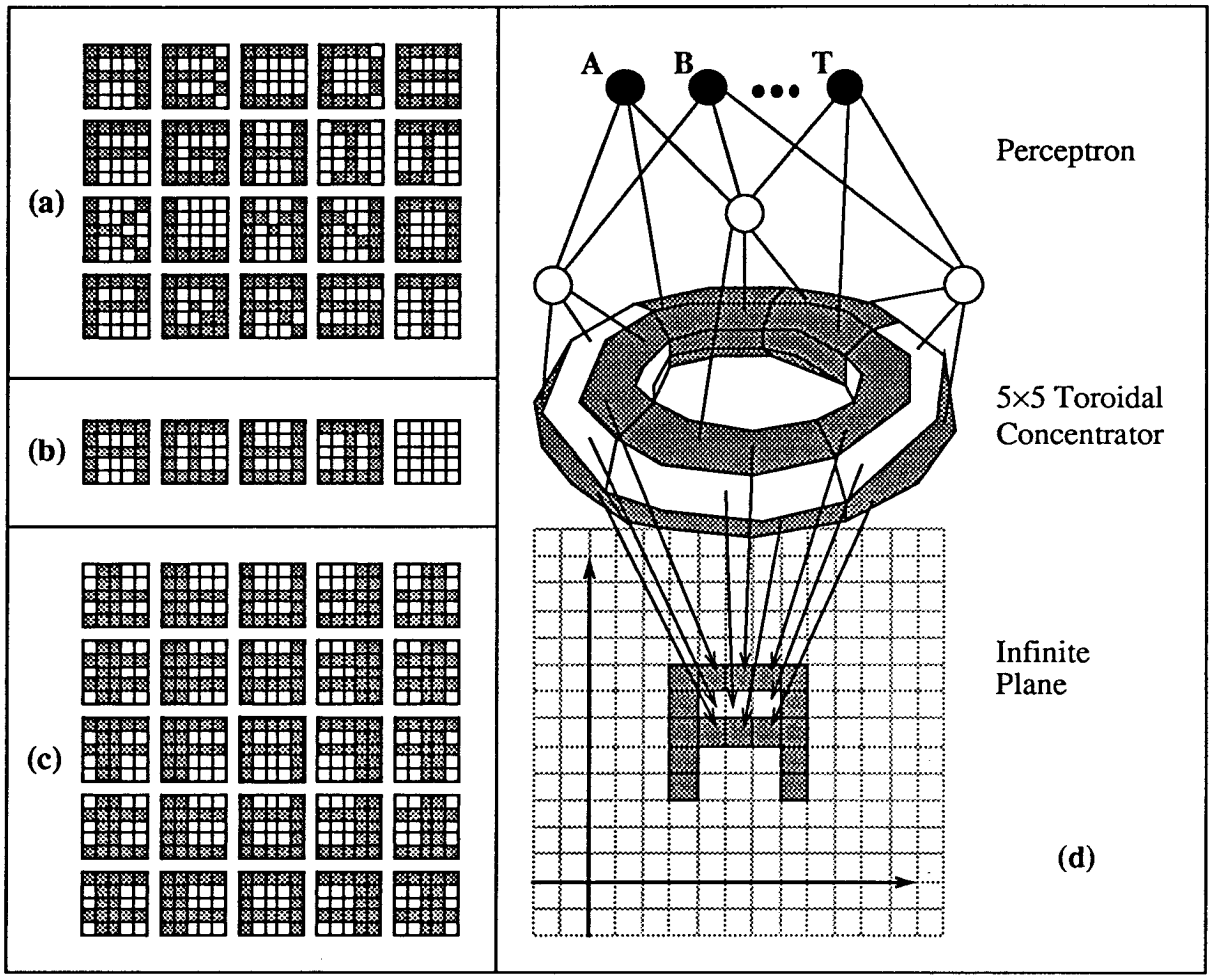


Fig. 5 - Experimental data for the perceptron construction
(a) character set, (b) rotations, (c) shifted characters, (d) toroidal transformation

A real perceptron able to classify all these patterns with 100% accuracy had ~9120 connections, 374 hidden units (127 of the 2-nd order, 118 of the 3-rd order, 124 of the 4-th order and 5 of the 5-th order) and weights ranging from -1.5 to 0.9. Its generation took 2117s on the VAX 3600.

3 Non-deterministic classification by perceptrons with real coefficients

This experiment aimed at verifying the performance of our learning algorithm in noisy environments. A model of a simple digit display was chosen to generate data for this series of experiments. Apart from the model's simplicity, the choice was dictated by two additional reasons:

(i) the model can be solved analytically, and
(ii) the same model was used earlier by Breiman *et al.* (1984) to demonstrate different features of their decision tree generating algorithm - CART.

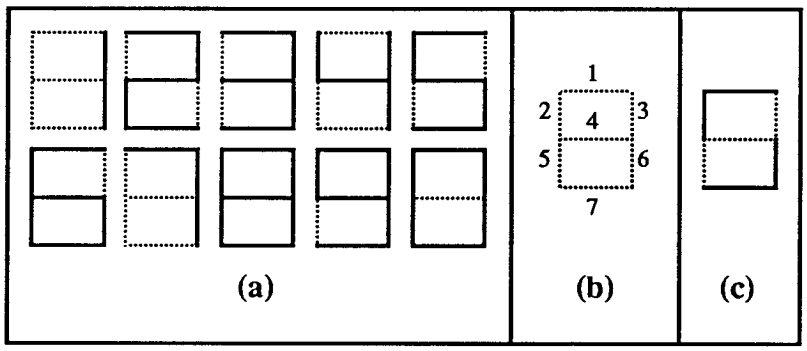


Fig. 6 - LED data set
(a) digit set, (b) character features, (c) ambiguous character

3.1 Background. The model considers ordinary LED displayed digits using seven horizontal and vertical lights in on-off combination (Fig. 6.a).

It is assumed that the display is faulty and each of the seven lights has probability 0.1 of not doing what it is supposed to do. Thus if, say, 5 is supposed to be displayed then any one of 2^7=128 possible patterns could be seen. In particular we can have the proper pattern for 5 with probability 0.9^7=.478, the pattern in Fig. 6.c with probability $0.9^6 \times 0.1 = 0.053$, etc.

If the pattern in Fig 6.c is displayed we can find that probabilities of display are: 0.728 for 5, 0.081 for 6, 9 and 0, 0.0090 for 3, 7 and 8, 0.0010 for 1 and 4 and 0.0011 for 2. The aim is to build a simple decision operator which allocates one of the classes 1, 2, 3,..., 9, 0 to patterns like in Fig. 6.c. in such a way that misclassification is minimal. It is known that Bayes decision rule allocating the class with the highest probability (e.g. class 5 for the pattern in Fig. 6.c) is optimal here. It can be proved that for this example the Bayes rule misclassification is ~0.26.

In practice we are forced to find simple suboptimal classifiers. A simple binary tree with 9 non-terminal nodes on p. 47 in Breiman, *et al.* , is an example of such a suboptimal classifier. Its misclassification rate estimated by a test sample of 5000 was reported to be ~0.30. The tree was initially grown to classify in the best possible way the first 200 exemplars of the test sample and then pruned appropriately to minimise the misclassification rate for the whole test sample.

In our experiments we tried to follow Breiman, *et al.* as closely as possible. We generated a test sample of 5000 using a standard random number generator. The sample was then tested by the simulated CART-generated decision tree with misclassification 0.301, which is very close to the Breiman, *et al.*, result. Then we used the first 200 exemplars to grow a number of perceptrons which subsequently were tested for misclassification estimated by the test sample of 5000. Finally we used some retraining procedures to improve performance of the best of our perceptrons to the theoretical level (~0.26).

3.2 Initial perceptron growing. Two families of real perceptrons were considered (cf. Fig. 7 and 8). Perceptrons in both families have 7 inputs representing LED lights and 10 outputs,

corresponding to digit classes 1, 2,..., 9, 0. For a given binary input pattern the output class with the largest of sums was selected as the final perceptron choice. Each of the perceptrons in the families was constructed in order to improve the mean-square-error (m.s.e) of its predecessor, measured for the first 200 exemplars (on the level of output sums). The improvement in each case was possible by lowering an *acceptance constant*, thus allowing new

	Terms used (input and hidden units)	No. of terms	Accept-ance	m.s.e. (200)	Misclassification rate			
					200	1000	2000	5000
1	1, 2, 3, 4, 5	5	0.100	0.599	0.252	0.305	0.313	0.299
2	+6∧7	6	0.095	0.564	0.235	0.280	0.295	0.287
3	+4∧5∧6	7	0.080	0.508	0.213	0.258	0.271	0.269
4	+3∧4∧5	8	0.060	0.474	0.232	0.274	0.283	0.280
5	+1∧2, 3∧5∧6	10	0.055	0.408	0.205	0.269	0.288	0.289
6	+1∧3∧4	11	0.050	0.388	0.210	0.266	0.283	0.285
7	+1∧6, 2∧3∧4	13	0.040	0.360	0.195	0.266	0.283	0.284
8	+2∧3∧7, 1∧2∧5∧7	15	0.035	0.338	0.200	0.270	0.286	0.285
9	+1∧4∧6	16	0.030	0.328	0.185	0.270	0.288	0.206
10	+1∧2∧4	17	0.025	0.318	0.183	0.274	0.293	0.296

Fig. 7 - 1st family of real perceptrons (+ means addition of new terms)

	Terms used (input and hidden units)	No. of terms	Accept-ance	m.s.e. (200)	Misclassification rate			
					200	1000	2000	5000
1	1, 2, 3, 4, 5	5	0.100	0.599	0.252	0.305	0.313	0.299
2	+6, 7, 1∧2, 2∧4, 3∧5, 1∧3∧4, 3∧4∧5∧6	12	0.050	0.380	0.205	0.271	0.285	0.276
3	+2∧5, 1∧5∧7, 1∧6∧7,2∧3∧7	16	0.033	0.332	0.190	0.264	0.275	0.269
4	+3∧4	17	0.025	0.319	0.185	0.269	0.283	0.280
5	+4∧5, 1∧3∧7, 2∧3∧6, 3∧6∧7, 1∧2∧3∧4∧5	22	0.020	0.289	0.165	0.265	0.285	0.285
6	+1∧3∧6	23	0.017	0.285	0.165	0.263	0.283	0.283
7	+1∧2∧3∧4, 2∧4∧6∧7, 2∧4∧5∧7, 5∧6∧7, 1∧4∧6,	28	0.014	0.265	0.150	0.270	0.293	0.294
8	+3∧4∧7	29	0.013	0.263	0.150	0.269	0.290	0.294

Fig. 8 - 2nd family of real perceptrons (+ means addition of new terms))

	Input & Hidden Units	Class of output unit									
		1	2	3	4	5	6	7	8	9	0
1	const	0.600	0.020	-0.042	0.144	0.326	0.335	0.278	-0.279	-0.246	-0.135
2	1	0.330	0.112	0.073 (0.097)	-0.279	0.050	-0.011	0.246 (0.045)	0.059	0.143	-0.062
3	2	-0.156 (-0.266)	-0.140 (-0.188)	-0.270	0.250 (0.219)	0.008	0.006 (0.004)	-0.155	0.018	0.233	0.205
4	3	-0.047	-0.035	0.094	0.073	-0.386	-0.350	0.022	0.254	0.201	0.173
5	4	-0.162	0.204 (0.171)	0.192	0.142	0.089	-0.002	-0.281	0.006	0.014	-0.202
6	5	-0.082	0.422 (0.374)	-0.107	-0.076	-0.084	0.039 (0.093)	-0.233	-0.027	-0.215	0.364
7	6∧7	-0.011	-0.208	0.196	-0.167	0.079	-0.090	-0.146	0.083	0.048	0.216
8	4∧5∧6	0.106	-0.420	-0.067	-0.051	-0.208	0.310	0.275	0.448	-0.008	-0.386

Fig. 9 - Selected perceptron (optimal weights are in brackets)

units to be added. (The *acceptance constant* can be viewed as a measure of the degree to which the m.s.e. may be reduced by addition of new units.) The misclassification rates for all these perceptrons, for the first 200 exemplars and then for the whole 5000, are given in Figures 7 and 8. We observe that although the m.s.e. decreased systematically with the higher complexity of perceptrons, the misclassification rate for 5000 has well pronounced minima regions on the level of relatively simple perceptrons. This phenomena can be interpreted as, that from a certain moment, with the increased perceptron complexity we try to adjust to 'the noise' in the sample of 200 rather than to 'the information'. Note that similar observations were made on numerous occasions in the past, e.g., in the area for decision trees, where they were used as a basis for the pruning algorithms (cf.. Breiman *et al.* 1984, Quinlan 1983). Note also that all of the above perceptrons have performance not worse than the final, pruned decision tree produced by CART.

Figure 9 lists weights for the best of these perceptrons (No.3 in Fig. 8). Note that each of the input units and each of the hidden units are connected to all output units, and that all weights, apart from output bias, have absolute values less than 0.4. Other perceptrons display similar features.

3.3 Retraining. In the test for 5000 exemplars (Fig. 9), even the best of our perceptrons had a misclassification rate higher than the theoretical optimum (0.26). The natural question arises whether it is possible to retrain (redesign) this perceptron, by changing some of its weights, in order to improve the performance. As a retraining algorithm one can use here for instance the Widrow and Hoff (1960) training algorithm and change the weights gradually. This procedure can be viewed as a simplified version of the popular back-propagation algorithm by Rumelhart, Hinton and Williams (1986) and, as experience shows, will be relatively

slow. However, it is not difficult to notice that in this particular case such an approach converges and leads to the globally optimal solution.

The approach to retraining we have taken, was different from that described above. We simply looked at the performance of the perceptrons all 128 possible input patterns and then we modified some of the weights 'manually' to force the perceptrons to be Bayesian classifiers. By changing selected weights of the perceptron in Fig. 9 to the ones in brackets, we obtained near optimal perceptron with misclassification rate for 5000 equal to 0.261. Furthermore, using for the retraining of perceptron No. 2 in Fig. 8 some 'true' statistics for the sample of 5000, differing from the theoretical ones, we were able even to lower the misclassification for 5000 to 0.2576, i.e. below the theoretical misclassification!

What this experiment shows is that it is still possible to improve the perceptron's misclassification rate by its further retraining. The whole process can be actually fully automated, which will extend it in future to more complex cases. However, before doing that we plan to evaluate a number of alternative approaches, some of which (like 'back-propagation') are 'mechanised' by their very nature.

3.4 Weights rounding. The purpose of the weights rounding is opposite to retraining: we aim here at 'technological' simplification of the perceptron structure at a cost of (hopefully insignificant) deterioration of performance. Apart from easing the manufacturing tolerance, the rounding affects the important ratio between the largest and smallest of weights (cf. the discussion of the scale problem in Epilogue to 1988 edition of Minsky and Papert). We found in a number of experiments that perceptron structural simplification by weight rounding was possible with marginal deterioration of performance. For instance, for the perceptron No.3 in Fig. 8 (cf. Fig. 9) we have the following. All its weights (Fig. 9) are originally specified to the third decimal place which, we recall, gives misclassification for 5000 equal to 0.2688. From our simulation it results that rounding weights in Fig. 9 to the second and to the first decimal place deteriorates misclassification rate for 5000 to 0.2694 and 0.2836, respectively. Note also that the ratios between the largest to smallest of weights are 300, 60 and 6 for the three, two and one decimal place weights, respectively. Thus the latter versions of perceptron are very attractive from the point of view of implementation.

Category	Class	ID
verb	infinitive	0
	present	1
	past	2
	present participle	3
	past participle	4
	special	5
adjective		6
adverb	simple	7
	complex	8
noun	plural	9
	singular	10
pronoun	demonstrative	11
	personal	12
	quantifier	13
	relative	14
	wh-question	15
other	article	16
	cardinal	17
	ordinal	18
	conjunction	19
	preposition	20
	proper nouns	21

Fig. 10 - Word lexical classes

Word	Classification (by IDs) 01234567890123456789 01	Comment
abolish	10000000000000000000	+ infinitive_verb
abolished	00101000000000000000	+ past_verb + pastpart_verb
abolishes	01000000000000000000	+ present_verb
abolishing	00010000000000000000	+ prespart_verb
abolition	00000000001000000000	+ singular_noun
abolitions	00000000010000000000	+ plural_noun
absolute	00000010000000000000	+ adjective
absolutely	00000001000000000000	+ adverb
abstract	10000010001000000000	+ adjective + infinitive_verb + singular_noun
abstracted	00101000000000000000	+ past_verb + pastpart_verb
abstractly	00000001000000000000	+ adverb
abstracts	01000000010000000000	+ plural_noun + present_verb
any	00000000000001000000	+ quantifier
anybody	00000000000010000000	+ pronoun
anyhow	00000000000000000010	+ preposition

Fig. 11 - Sample words classification

4 Deterministic binary perceptrons

The aim of this experiment was to construct and retrain a very large binary perceptron which could be used as an information retrieval system. The application chosen was that of lexical classification of English words, and the constructed network was subsequently used as a part of a massively parallel integrated natural language parser COSIMO (Jennings, Rowles and Kowalczyk 1989, Cybulski and Jennings 1989, Cybulski, Kowalczyk and Szymanski 1989b).

COSIMO's lexicon consists of 1000 most frequently used English words of up to 10 characters, classified into 22 lexical categories (cf. Fig 10). Some of the lexicon words fall into multiple classes, e.g. *abstract* may be classified as an *adjective*, a *noun* and an *infinitive verb*. Thus, a lexicon may be viewed as a mapping from a set of words into a set of lexical categories and classes. Since the mapping function is purely deterministic and a word and its classification may easily be represented with the strings of binary values, the natural choice for the lexical encoder was a binary perceptron.

Each of the perceptron training rules consisted of the word and its classification (Fig. 11). The word bitmap constituted a perceptron input layer of 260 units representing 10 characters encoded as a series of 26 bits, each of the bits corresponding to one character of the English alphabet. An ASCII encoding was also considered but experiments proved that a more sparse word representation on input results in reduced interconnectedness in hidden

and output layers of the perceptron. The output layer consisted of 22 units, each corresponding to one of the lexical classes.

The perceptron training required 4138 s of CPU time on the VAX 3600 and aimed at 100% accurate word classification (for words embedded in the training set). The generated perceptron had a mere 734 hidden units of 2-nd order, 1 unit of 3-rd order, and 8267 synapses. The simulations of the perceptron-based word classifier showed the system to be rather slow, nevertheless, its parallel architecture required only two levels of AND binary gates for the implementation of a hidden layer and 8 levels of XOR binary gates for the outputs, and thus its hardware implementation could be very efficient - 10 clock cycles per classification in synchronized architecture, or even faster in unsynchronized implementation.

It should be noted that the lexicon implementation in a real or integer perceptron would require fewer hidden units and layer interconnections, however, the storage requirements for the binary perceptron are still drastically smaller (1 bit units as opposed to 16 or 48 bit units).

5 Conclusions

This paper introduced three different classes of two-layer perceptrons :- real, integer and binary. It was suggested that some of the fully distributed learning methods, like back-propagation, may be enhanced by some global pre-initialization of the perceptron architecture. The training algorithm for the perceptron global optimization was briefly discussed and several test results were reported, in particular:

(i) It was shown that perceptrons with real and integer coefficients are able to handle efficiently deterministic problems requiring large sets of exemplars (500 cases, 25 inputs, 20 outputs, and hidden units of 5-th order).

(ii) Real perceptrons are well suited for solving problems in noisy environment. We were capable of designing a simple perceptron classifier with performance close to optimal (i.e. theoretical misclassification - 0.26, misclassification decision tree designed by CART - 0.301, ours - 0.261).

(iii) In an attempt to apply perceptrons to the information retrieval problems, we found that binary perceptrons are quite effective in dealing with reasonably large volumes of data (1000 records, 260 bit keys and 22 bits of data), and that their hardware implementation could lead to very fast response times (10 machine cycles).

(iv) In large deterministic problems, the number of hidden units is of the order of the number of exemplars.

(v) CPU time increases approximately proportionally with the size of the training set raised to the power equal to the highest order of the required hidden units.

(vi) The simplicity is higher and the order much lower for the computer generated perceptrons as compared to the human generated solutions.

The presented results show that perceptrons have a great potential in the area of pattern recognition, classification, information retrieval, decision making in noisy environments, but also in the area of parallel hardware design. Hence, it is reasonable to expect that perceptrons could enhance or even replace other traditional methods of decision making, e.g. decision tables and trees.

6 Acknowledgement

We acknowledge the permission of Executive General Manager, Telecom Australia Research Laboratories to publish this paper.

7 References

Ackley, D.H., Hinton, G.E. and Sejnowski, T.J. (1985): "A Learning Algorithm for Boltzmann Machines," *Cognitive Science*, Vol 9, pp 147-169.

Breinman, L., Friedman, J.H., Olshen, R.A. and Stone, C.J. (1984): *Classification and Regression Trees*, Belmont, California: Wadsworth International Group.

Cybulski, J.L., Jennings, A. (1989): "COSIMO: A Massively Parallel Integrated Parser" CSS Branch Paper 183, Telecom Australia.

Cybulski, J.L., Ferrá, H.L., Kowalczyk, A. and Szymański, J. (1989): "Determining Word Lexical Categories with a Multi-Layer Binary Perceptron" CSS Branch Paper 182, Telecom Australia.

Grossberg, S. (1980): "How Does a Brain Build a Cognitive Code?," *Psychological Review*, Vol 87, pp 1-51.

Hopfield, J.J. (1982): "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," *Proc. of the National Academy of Sciences*, Vol 79, pp 2554-2558.

Jennings, A. , Rowles, C. D and Kowalczyk, A. (1988): "Natural Language Understanding in the MEDICI Project," *Proc. Int. Conf. on Comp. and Inf.*, Toronto, Canada 1988.

Kowalczyk, A. (1989a): "Empirical Induction Algorithm for Approximate Reasoning" CSS Branch Paper 178, Telecom Australia.

Kowalczyk, A. (1989b): "Optimisation of Decision Operator" CSS Branch Paper 179, Telecom Australia.

Lippmann, R.P. (1987): "An Introduction to Computing with Neural Nets," *IEEE ASSP Magazine*, IEEE, April, pp 4-22.

Michalski, R.S., Carbonell, J.G. and Mitchell, T.M. (1983): *Machine Learning: An Artificial*

Intelligence Approach, Palo Alto, California: Morgan Kaufmann Pub. Inc.

Michie, D. (ed. 1979): *Machine Intelligence*, Edinburgh: Edinburgh University Press.

Minsky, M.L. and Pappert, S.A. (1969): *Perceptrons*, Cambridge, Massachusetts: The MIT Press. (Expanded edition, 1988.)

Pawlak, Z., Wong, S.K.M. and Ziarko, W. (1987): "Rough Sets: Probablistic Versus Deterministic Approach," Technical Report, Department of Computer Science, University of Regina.

Quinlan, J.R. (1979): "Discovering Rules from Large Collection of Examples: A Case Study," *Expert Systems in the Microelectronic Age*, in Michie 1979.

Quinlan, J.R. (1983): "Learning Efficient Classification Procedures and Their Application to Chess and Games," in Michalski, Carbonell and Mitchell 1983, pp 463-482.

Rumelhart, D.E., Hinton, G.E. and Williams, R.J. (1986): "Learning Internal Representations by Error Propagation," in Rumelhart and McClelland 1986, pp 318-362.

Rumelhart, D.E. and McClelland, J.L. (eds 1986): *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume1: Foundations*, Cambridge, Massachusetts: The MIT Press.

Widrow, B. and Hoff, M.E. (1960): "Adaptive Switching Circuits," *1960 IRE WESCON Convention Record*, New York: IRE, pp 96-104.