Feedforward Networks: from Theory to Technology*

P. Ruján
Fachbereich 8 Physik and ICBM
Carl-von-Ossietzky Universität
Postfach 2503, D-2900 Oldenburg

1 Introduction

At the origin of this pretentious title was my desire to convey the feeling that neuronal network-type computing is slowly becoming a full fledged technology. The theory (at least for feedforward networks) is in reasonable shape, especially when considering the complexity of the problem. A lot of efforts have been invested in developing fast algorithms for learning and there are now some well behaved basic methods complementing the well-known back-propagation algorithm[6]. On the hardware front, besides already available analogue chips, a few digital special processors for fast matrix-vector operations are announced for this year, allowing a reasonable size network making decisions at around 10 $msec$ rate.

Due to space limitations, I will concentrate on the problem considered in collaboration with the MPI München group[1], the implementation of a second level neuronal network trigger at HERA. The problem was to develop a classifier which distinguishes between physical and wall-gas background events. From earlier simulations[2] it has been concluded that each type of physical event is best separated by its own neural network. Furthermore, from the Monte Carlo data used to construct the classifier, it became clear that for almost all separations one single neuron (or Perceptron) is enough. One notable exception was the $c - \bar{c}$ type events: using all 23 input channels, the separation of $c - \bar{c}$ events from background has been the most difficult one, as expected also on theoretical grounds. We decided to compare different learning methods on this particular problem. The example set consisted of 22675 background and 3390 $c - \bar{c}$ events, of which the first 80% were used as training and the rest as test set.

The L_2 trigger will be built in hardware: each input channel contains some energy or momentum data in digitalized 8 bit format — for more details see the previous paper. There are several options for the chips to be used: thus the accuracy of a weight and threshold might be either 8 or 16 bits, depending on the actual choice. The number of hidden units is limited to 16 or 64 units. Thus, this problem poses some interesting technological constraints on the learning algorithms. The results obtained by the LUND-network program JETNET 2.0 (an improved back-propagation-like stochastic descent) were compared to the results of a constructive method[10]. I what follows I will explain how technological constraints of this kind can be imposed and what each hidden unit is supposed to do in the latter approach.

*Proceedings of the Second International Workshop on Software Engineering, Artificial Intelligence and Expert Systems in High Energy and Nuclear Physics, l'Agelonde France-Télécom, 1992

2 Learning from Examples in Feedforward Networks

I shall consider exclusively McCullogh – Pitts[3] binary neurons, which are activated only when the weighed sum of their inputs is larger than some threshold:

$$\sigma = \mathrm{sgn}\left(\sum_i w_i s_i - \Theta\right) \tag{1}$$

where the vector $\vec{w} = (w_1, \ldots w_N)$ is a vector of weights representing the strengths of impinging synapses from N other neurons whose activation state is denoted by $(s_1, \ldots, s_N)$, $s_i = \pm 1$. Θ is a threshold value. The output of the processing unit is $\sigma = 1$ if the 'neuron' fires a high frequency train of spikes or $\sigma = -1$ if it remains inactivated. According to Eq. (1) every elementary processor classifies the input signal as being in one or another side of the N-dimensional hyperplane

$$\vec{w}\vec{s} = \Theta \tag{2}$$

The terms 'hyperplane' and 'unit' can be thus interchanged when associating the vector $\vec{w}$ and the threshold Θ with unit σ.

An architecture with a single output processor unit is a linear classifier known as a *Perceptron* [4]. Perceptrons can represent only the class of 'linearly separable' boolean functions and many practical tasks fall outside this class [5]. Adding one or more layers of so-called hidden units without allowing feedback connections leads to a multilayer feedforward architecture. Such devices can implement arbitrary boolean functions at the price of using most of the times an exponentially large number of hidden units. Learning from examples means in this context searching in the space of all weights and thresholds the parameters minimizing some error function defined on the training and/or the test set.

Unfortunately, this search is $\mathcal{NP}$-complete [7] already for two hidden units [8]. From a 'physical' point of view this means that the search is hindered by an exponential number of local (metastable) minima and has consequently a very slow dynamics. The only way to 'escape' the unpitiful laws of algorithmic complexity is to change the rules of the game. As we shall see, this does not mean necessarily cheating. One way of proceeding is to start without any preconception about the structure of the network and to construct it as to best fit the set of examples (nonlinear, nonparametric estimation).

3 Information Processing

Consider the situation shown in Fig. 1. Points represent (example)vectors in the input configuration space and their color denotes the desired output value (black for +1 and white for -1). The first layer of hidden units (hyperplanes) will partition the configuration space of input vectors into several regions. In an ideal situation, all points lying into a given region will be mapped into a single point (vector) of the first hidden layer configuration space. This map contracts since it is a many-to-one map. Obviously, errors occur when points of different colors are mapped into the same point. (Sometimes, we might WANT to include a few errors in a controlled way. This is the case for the trigger problem, since the examples are generated from overlapping distributions.) Hence, a necessary condition for an error free training is that the partition of the configuration space with hyperplanes results on regions populated by the same kind (class) of examples. An additional benefit of this simple picture is that it hints at the following trivial theorem: if a constructive algorithm defines in every step (from one layer to another) a contractive map, then it will necessarily converge. This is the basic idea for proving convergence in most network growth algorithms.

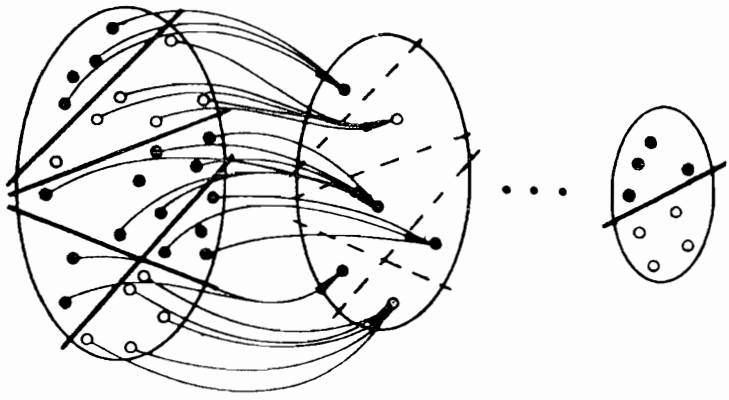

Figure 1: *Information processing in feedforward networks a) The input configuration space is partitioned in several regions by the hidden units of the first layer b) The first hidden unit configuration space: every region of the previous input space is mapped into a single point c) The last hidden layer defines a linearly separable function. The black and white points are separated by the output unit plane*

.

4 Sequential Learning

A little bit more ambitious question is how to partition the *last* hidden layer so as to *guarantee* linear separability. Answering this question allows one to consider — without loss of generality — only architectures with one single hidden layer. From the point of view of execution time, this is optimal. Other considerations, like the lack of enough processing units, might favor a many-layer solution.

We obtained two sufficient conditions for linear separability of a *hidden* layer. One is based on a geometrical construction[9], the other one is a variant of the stratification method[10]. I will only state the second condition, used for the trigger learning algorithm.

A *sequential learning procedure* is defined as following[10]: Assume one has some method of searching for hyperplanes (alias hidden units) such that on one side of the hyperplane there are (possibly many) points of the same class, σ, only. The actual set of training examples is said to be *cut* with such a plane by removing those examples from the training set. Assume now that after H such steps the remaining examples are linearly separable. Then H hyperplanes (hidden units) partition the input configuration space in at least $H + 1$ boxes. In [10] a constructive proof is given that a any sequential partition of the configuration space yields a linearly separable hidden space.

5 The Trigger Problem: Algorithm and Results

The Monte Carlo data show that the distribution of the background and of the $c - \bar{c}$ physics overlaps. In such situations no perfect classifier exists: a theoretical lower bound of the error is given by the Bayes criterion[11] (see Fig. 2).

The Bayes discriminant surface BDS is defined by

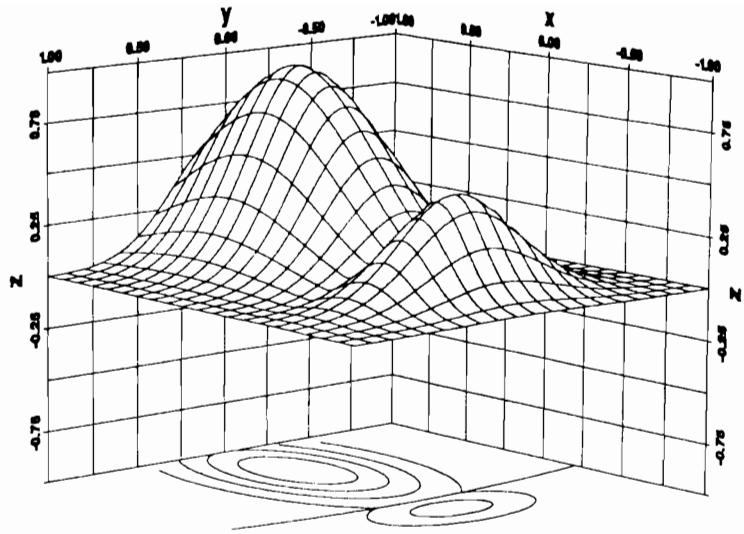

Figure 2: *The Bayes criterion in two dimensions. The left distribution corresponds to* $p(\vec{r}|class = back)P(back)$*, the right distribution to* $p(\vec{r}|class = phys)P(phys)$*.*

$$BDS = \{\vec{r} : p(\vec{r}|back)P(back) = p(\vec{r}|phys)P(phys)\}$$

where $P(back)$ and $P(phys)$ represents the *a priori* probability for a background and for a physics event, respectively. The minimal theoretical error is defined as

$$MinErr = \int_{\vec{r}\in Phys} dV\, p(\vec{r}|back)P(back) + \int_{\vec{r}\in Back} dV\, p(\vec{r}|phys)P(phys) \tag{3}$$

where BDS partitions the space in two (or more) subpaces denoted by $Phys$ and $Back$, respectively. Note that the Bayes discriminant surface is in general not linear but can be arbitrarily approximated by a piecewise linear surface, corresponding to (many) hyperplanes. Since in operational mode $P(back) \sim 10^5 P(phys)$ for the L_2 trigger, the best possible strategy is to cut **as much background as possible**, even at the expense of decreasing the luminosity. In fact, our results indicate that it is possible to cut almost all the background at the price of losing $40 - 50\%$ of the $c - \bar{c}$ physics events.

The algorithm used for the trigger problem is extremely simple: assume one generates at random a direction vector $\vec{w}$ in the 23-dimensional input space. Along each direction one constructs the map, the projection of all examples onto the direction $\vec{w}$:

$$h_\nu = \vec{w}\vec{\xi}^{(\nu)} \tag{4}$$

where $\{\vec{\xi}^{(\nu)}\}$ denotes the set of examples. After ordering the fields h_ν in ascending order, the two 'ends' of this map are analysed. Note that at each location h_ν the class and the index of the projected example is known.

Next, let us define a given proportion of allowed mistakes in classifying physics events, $err(phys) = 0.002$, for example. Starting from the maximal (minimal) h_ν value, we proceed downwards (upwards) until the number of 'bad' points does not increase over say, twice the allowed value (err is given in form of a ratio of 'bad' points over 'total' number of points). By exceeding the allowed rate of error by a factor of two or so helps to jump over possible local fluctuations of this ratio along the direction $\vec{w}$.. By storing the best case when the number of mistakes was still equal to the allowed bound, we can associate with each direction two possible cuts. We keep always the deepest cut in store and after repeating the procedure around 6000

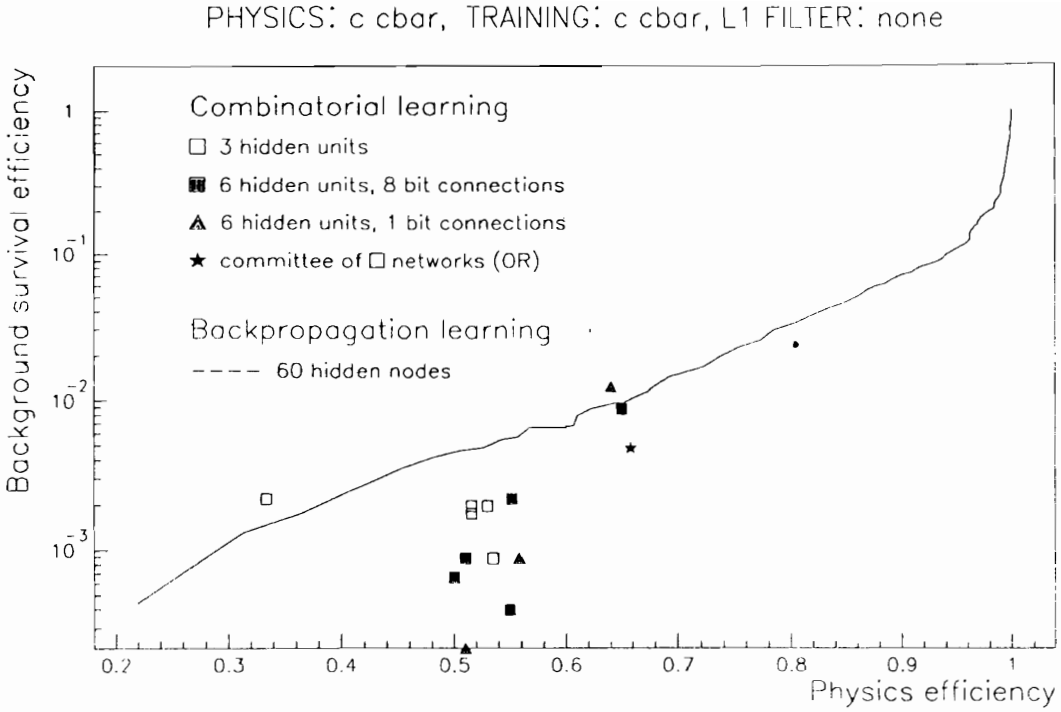

Figure 3: *Comparison between different trigger networks. See text for explanations.*

or so times, one takes the deepest cut. Both 'bad' and 'good' points lying on the cut side of the hyperplane are removed from the list of examples. The procedure is sequentially repeated until on both sides of a plane the conditions on errors are fulfilled (corresponding to 'linear separability') or when the maximally allowed number of hidden units has been reached. Last, the output unit is calculated using the recursion given in [10].

Since the directions are generated at random, it is easy to enforce technological constraints on the weight vectors. Similarly, the number of hidden units can be changed at will. Fig. 3 shows how this extremely simple procedure compares to the *best* back-prop network.

Fig. 3 shows the results of preliminary runs for different weight accuracy, different hidden unit numbers, and different learning algorithms. The naive Monte Carlo algorithm described above was used to generate the networks described by filled squares (6 hidden units, 8 bit accuracy for each weight), and triangles (6 hidden units, 1 bit $w_i = \{0, 1\}$ weights). Note that one of these networks made not a single mistake in removing the test background events. Results for similar networks with $w_i\{-1, 1\}$ are significantly worse. The original method[10] has been used for the networks with three hidden units (no restriction on weights) indicated by open squares. The star denoted network was obtained by combining *all* 3 hidden units networks with an AND gate, so as to improve physics efficiency. The learning time for obtaining a network is around 40 min. CPU on a SPARC-SLC Sun station and increases typically as $O(MN^2N_h)$, where M is the number of examples, N is the dimension of the input, and N_h the number of hidden units.

The continuous line represents the best network found by the MPI group (previous paper) with the LUND Jetnet version 2.0 program, an improved back-prop implementation for high energy physics. The network has 60 hidden units. Note that the vertical scale is logarithmic and thus several of our networks represent a background elimination efficiency several order of magnitude better than backprop.

The success of the naive Monte Carlo method is related to the fact that the background is homogeneously and compactly distributed, as expected. Hence, even when using random directions, one still finds some portion of the Bayes discriminant surface. The relatively big scatter on the efficiency of the networks is due largely to the fact that at this accuracy, a difference in 2 or 3 missclassified examples make for a relatively large difference. Also, the generalization probability should be more accurately calculated using a cross validation procedure.

Very recently, P. Ribarics[12] and collaborators have obtained comparable to better results

after increasing significantly the number of training epochs in the Jetnet algorithm. Their best network has now 25 hidden units. However, in view of the simplicity and learning speed of the constructive algorithm described above, it seems worthwhile to further investigate this possibility by improving the search strategies using more sophisticated Monte Carlo methods.

References

[1] A. Gruber, C. Kiesling and P. Ribarics — previous paper in this Volume

[2] J. Fent, C. Kiesling and P. Ribarics H1-MPI-150, 23.4.1991

[3] McCullogh W. S. and Pitts W. (1943) *Bull. Math. Biophys.* **5** 115-133

[4] Rosenblatt, F. *Psychoanalytic Review* **65** (1958) 386

[5] M. L. Minsky, S. Papert: *Perceptrons: An Introduction to Computational Geometry* Cambridge Ma., MIT Press, 1969 and 1988

[6] D. E. Rumelhart, G. E. Hinton, R. J. Williams: *Nature* **323**, 533-536 (1986)

[7] Judd S. *Proc. IEEE First Conference on Neural Networks San Diego 1987* Vol. II pp. 685-692 (IEEE Cat. No. 87TH0191-7)

[8] Blum A. and Rivest R. *Advances in neural information processing systems* pp. 494-501 Touretsky D. (Ed.) Morgan Kaufman, San Mateo, Ca., 1988

[9] Ruján P. and Marchand M. *Complex Systems* **3** (1989) 229-242 , *Proceedings of IJCNN 1989, Washington* Vol. II 105-110

[10] Marchand, M., Golea, M. and P. Ruján *Europhys. Lett.* **11** (1990) 487-492

[11] Duda R.O. and Hart, P.E.: Pattern Classification and Scene Analysis J. Wiley and Sons, New York, 1973

[12] P. Ribarics: private communication