Nuclear Instruments and Methods in Physics Research A317 (1992) 323-334
North-Holland

NUCLEAR INSTRUMENTS & METHODS IN PHYSICS RESEARCH Section A

Calorimeter trigger applications, training, and assessment of a feed-forward neural net

D.Y. Wu and M.K. Campbell

University of Michigan, Ann Arbor and SSC Laboratory, Dallas, TX, USA

Received 28 October 1991 and in revised form 14 January 1992

A feed-forward neural net technique is applied to trigger on $b \to e\bar{\nu}X$ decays in the high background environment of $\bar{p}p$ collisions at $E_{cm} = 1.8$ TeV. For the 1992 data run, we will install electronics into the CDF detector trigger using calorimeter signals as input to a neural net IC chip. We describe our algorithm and network optimization (training) issues, including those unique to $\bar{p}p$ physics and to an electronic implementation. We assess the advantages and problems of using trained neural nets. In addition to the trained mode of operation, the architecture of the analog chip also allows employing it as a massively parallel arithmetic processor for conventional trigger applications. The device may be reprogrammed to accommodate changes in trigger requirements and to provide considerable flexibility in trigger design.

1. Introduction

The large cross section (50 mb at $E_{cm} = 1.8$ TeV) from inelastic hadron collisions at $\bar{p}p$ accelerators makes it difficult to trigger on interesting processes which contain low energy decay products. This is especially true of events requiring calorimetry for identification, such as for $b \to e\bar{\nu}X$ decays. Good pattern recognition is needed to improve the signal-to-background ratio. Such pattern recognition requires analyzing and correlating critical subdetector signals, which is time consuming to process electronically and therefore typically implemented in secondary level triggers.

We will implement one particular type of pattern recognition technique based on neural net IC technology, new to triggering in high energy physics, in the CDF detector level 2 trigger for the 1992 data run at $E_{cm} = 1.8$ TeV. Neural networks are one method to correlate information in order to maximize the separation of signal from background.

Our neural network consists of a nonlinear function containing free parameters which must be optimized via a minimization procedure, known as training. There are several reasons for investigating the utility of neural nets. The network can be trained to distinguish very complex patterns. At the trigger level, one of the major assets of neural net ICs is their flexibility. The analog network chip may be reprogrammed to accommodate changes in trigger requirements. In addition, neural nets have fast parallel processing capabilities and are rather insensitive to fluctuations, such as from electronic noise or channel-to-channel differences in detector response. A final reason for pursuing neural nets is that powerful conventional cuts are sometimes difficult to implement electronically. The hope is to realize a similar signal-to-background ratio via a neural net cut for which the electronic capability exists.

Electronics manufacturers have produced IC chips, the 80170NX, which mimic the nonlinear function used in neural nets [3]. One downloads such a device with some set of optimized weights prior to trigger installation for data taking.

We provide an offline study of using a neural net to trigger on $b \to e\bar{\nu}X$ jets with calorimeter signals, a network application proposed by Campbell. We use the feed-forward multilayer (signals are fed forward through layers of processing nodes) network to separate the $b \to e\bar{\nu}X$ signal from gluon-jets background. An overview of the trigger, the input signals from the CDF calorimeter, and a brief explanation of our neural net are given in section 2. The expected efficiencies, the event kinematics, and some network issues relevant to triggering are considered here as well. A major aspect of employing neural nets is to first train it to recognize signal from background. Therefore, we describe our algorithm in detail as well as discuss the advantages and problems of using trained neural nets in the appendix.

There is a second application for neural net chip technology: the circuit architecture provides a unique flexibility that is valuable for analog trigger development, as suggested by Kuhlmann and Wu. For trigger design purposes, it is often known in advance which subdetector (e.g., calorimeter) signals would be useful

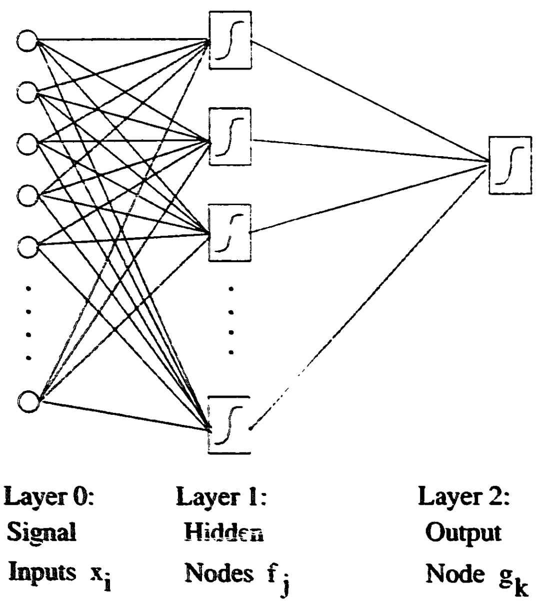

Fig. 1. A feed-forward neural net containing one layer of input signals, a hidden and an output layer. The processing nodes in the last two layers, represented by rectangles, consist of the sigmoid function described in the text. The lines connecting the inputs x_i and the hidden nodes multiply the x_i values by a factor w_i, and similarly for the lines between the hidden and output nodes. For the conventional trigger application, the nodes may be thought of as many parallel arithmetic processors.

in forming electronic decisions. However, what the best trigger algorithm for utilizing such signals prior to data taking at new energy or luminosity regions is less clear. It is possible to design general circuits using the 80170NX without having a fixed trigger algorithm in advance. The interconnections in the chip are such that the device may be used as an analog arithmetic processor to improve the performance of conventional trigger tasks, allowing reprogramming of threshold values and even revision of the type of arithmetic operations needed. For example, conventional electron and photon triggers involve threshold requirements and sums of calorimeter module energies. We try out this second way of using the 80170NX to form a conventional isolated photon trigger, which requires no training and is discussed in section 3.

Our neural set structure is shown in fig. 1 and described in section 2.1. Signals are input and processed (in the rectangular symbols) in parallel.

2. $b \rightarrow e\bar{\nu}X$ application of neural nets at $\bar{p}p$

Although our main goal is to test the utility of neural net technology for trigering, our physics interest lies in studying b's at $\bar{p}p$ where we hope to log more of the large 10–40 μb $b\bar{b}$ production cross section [2]. Measurements of rare B decays and $B\bar{B}$ mixing, for example, require as large a sample of b's as possible. We select $b \rightarrow e\bar{\nu}X$ jets as our signature for testing the new trigger and use a trained net to try to improve the signal-to-background ratio over that obtained with the conventional electron trigger. The cross section for b physics is about 10^3 times smaller than for the typical inelastic hadron collision ("minimum bias" event) and it decreases sharply as a function of the momentum of the b transverse to the beam axis (p_t). In order to trigger on more b jets, it is necessary to lower the energy thresholds. However, it becomes difficult to distinguish the B decay particles from the gluon-jet backgrounds at lower energies.

There are practical reasons for choosing to trigger on $b \rightarrow e\bar{\nu}X$ jets. For the CDF detector, calorimeter signals are available more quickly than track momentum information and therefore are better for forming a fast trigger. Such a calorimeter-based pattern motivates selecting a physics process involving electrons, such as $b \rightarrow e\bar{\nu}X$ decays. Furthermore, CDF has already collected $b \rightarrow e\bar{\nu}X$ data where the E_t of the electron is > 7 GeV, providing a useful data sample for trigger development. CDF will also implement a conventional electron trigger, allowing cross checks of the neural net $b \rightarrow e\bar{\nu}X$ results.

We now describe the CDF calorimeter [3] signals which are fed to the neural net. Cylindrically symmetric about the beam axis, the central calorimeters consist of towers which point back to the interaction region. Each tower has an electromagnetic shower counter (lead scintillator) in front of a corresponding hadron calorimeter (steel scintillator). The energy resolution for the central electromagnetic calorimeter is $\delta E/E \approx 0.14/\sqrt{E}$ (E in GeV), and for the hadronic calorimeter is $\delta E/E \approx 0.7/\sqrt{E}$. Calorimeter signals representing information in bins of 15° in azimuth ϕ by 0.2 in pseudorapidity ($\eta = -\ln(\tan \theta/2)$, where is θ is the polar angle) reach the trigger electronics [14]. The neural net $b \rightarrow e\nu X$ trigger obtains signals from a $14\eta \times 24\phi$ grid area with two layers in depth (electromagnetic and hadronic).

A $b \rightarrow e\bar{\nu}X$ jet where the b has > 15 GeV p_t would typically deposit all of its energy in 5×5 bins, subtending 37.5° by 0.5η from the center to the outer edge. The electron energy is normally deposited in a single 1×1 electromagnetic trigger tower which is 18 radiation lengths thick. The particles from the hadronized b quark jet deposit hadronic and electromagnetic energy in a few towers and the underlying minimum bias event normally deposits much less than 0.5 GeV per tower.

Our procedure for triggering on the $b \rightarrow e\bar{\nu}X$ decay is to look for central electromagnetic (CEM) calorimeter trigger towers that pass a certain E_t threshold (seed towers) [5]. The signals from the seed tower and from nearby towers (a 5×5 calorimeter grid region centered on the seed tower with coordinates (η_s, ϕ_s)) are fed to to the neural net electronics. There are altogether 50 trigger towers signals, half electromagnetic, half hadronic. The network then decides if the calorimeter pattern indicates the presence of a $b \to e\bar{\nu}X$ jet. After the neural net electronics finish processing, we require a charged track with p_t above some threshold to extrapolate to the seed tower in the ϕ direction, further indicating the presence of an electron.

2.1. A feed-forward multilayer network

The feed-forward 2 layer neural net we use to separate signal from background patterns is shown in fig. 1. The network consists of a nonlinear function, S, involving the product of the input values x_i (the 50 calorimeter signals) and of weight parameters w_i, and thresholds (biases) b_j:

$$S \equiv g_k\left(\sum_j \omega'_{jk} f_j\left(\sum_i \omega_{ij} x_i + b_j\right) + b_k\right),$$

where i is an index over the inputs, j the hidden nodes and k the output nodes (only one is shown). The two layers of functions f_j and g_k are of the same mathematical form, a sigmoid function (represented by the rectangles in fig. 1):

$$f(y) = \frac{1}{1 + e^{-y}}.$$

We obtain optimal weight and bias values for S via a minimization procedure which tries to separate signal from background patterns. We use patterns known to be either signal or background (e.g., from a Monte Carlo sample) and assign a desired response value "0" or "1" to the pattern during the minimization. The

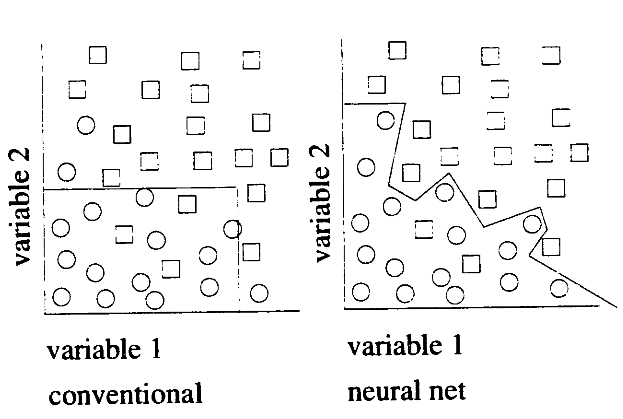

Fig. 2. A neural net requirement (left) tends to achieve beter signal-to-background separation than conventional, fixed-value cuts (right). Shown is a hypothetical case involving the distribution of signal (circles) and background (squares) events for two arbitrary kinematic variables.

process is essentially fitting the function S to a surface proportional to the relative amount of the two sets of patterns used during training such that the surface best separates signal from background. The appendix provides an explanation of the algorithm we use, which is adapted from Rumelhart and McClelland [6]. The procedure is changed slightly to account for the fact that there are many more possible patterns from the background class. We afterwards check the separation efficiency of the trained neural net by running it on an independent test sample.

Signal and backgrounds tend to show better separation using a neural net technique than with one-dimensional cuts. For any set of variables used to characterize events, the actual boundary of best separation between the signal and the background does not normally consist of flat surfaces corresponding to fixed values on each variable; the boundary has some curvature. Using conventional single-valued threshold cuts, which do not account for correlations between variables, leads to needless loss in signal or retention of background. The neural net training procedure finds the parameters of S such that contours of constant S approximate the actual curvature of the boundary, as shown in fig. 2 for a hypothetical case involving two variables. Appendix E gives a simple numerical example.

In order to understand the neural net solution after training, we need to examine physical quantities which may be formed from the 50 tower energies fed as inputs to the network. For example, fig. 3 provides the distribution of neighboring vs seed trigger tower electromagnetic energy for test patterns passing and failing the neural net. It is worth noting that neural net training can provide insights into the kinematic features of the physics process, useful for optimizing threshold algorithms for conventional triggers and analyses. For both signal and background, fig. 3 also demonstrates that a final neural net output threshold cut (cut on a particular value of S) does not necessarily correspond to a flat cut on the values of the input variables.

The very advantage of a neural net algorithm also leads to a possible problem of estimating trigger biases and efficiencies. One must understand the detector response over a large range of values for the input variables because the final network cut corresponds to a cut over a large range ("multi-valued" cut) of those input signal values, as shown in figs. 2 and 3. However, for most conventional, single-valued cuts on a particular variable, one needs to estimate the efficiency of the requirement only near the cut value. If Monte Carlo calculations are used either to train the network or to estimate efficiencies and errors, they must be correct over a broad range of values and correlations among the many input variables. Otherwise, it may be difficult

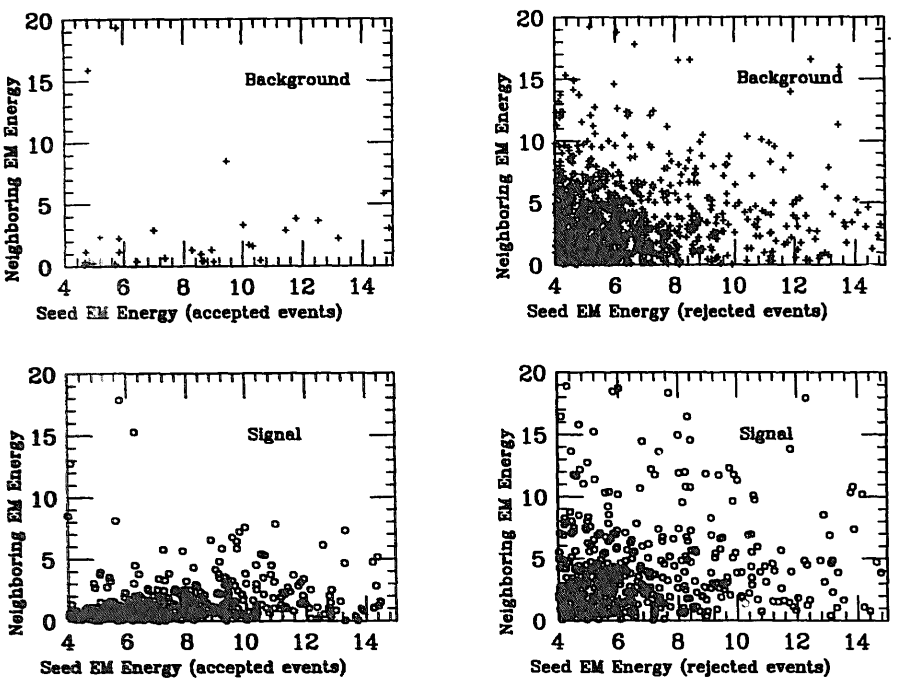

Fig. 3. The distribution of neighboring vs seed trigger tower electromagnetic energy for Monte Carlo signal and background test patterns accepted and rejected by the trained neural net. It is necessary to plot physical variables in order to understand the nature of the neural net cut.

to know if the final estimated efficiencies and systematic errors for a neural cut are reliable. This problem is reduced by running redundant and less stringent triggers along with the neural net trigger.

2.2. *Description of the training and the trigger rates*

The results from an offline simulation of the neural net trigger are considered in this section. We train on

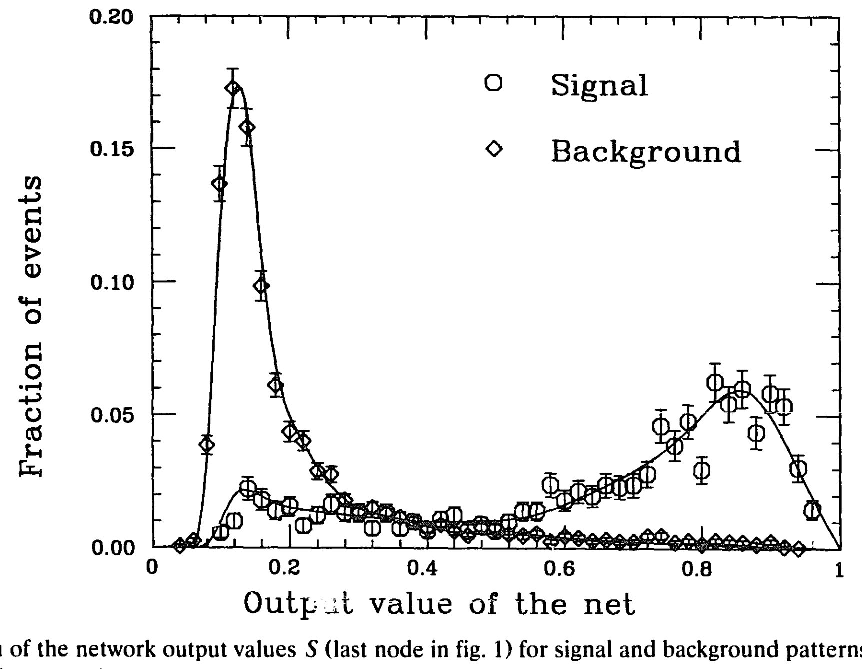

Fig. 4. The distribution of the network output values S (last node in fig. 1) for signal and background patterns, where the curves are normalized to unity. The exact signal-to-background ratio depends on our choice for the trigger threshold; we typically use a value in the range 0.7–0.8. When normalized to the estimated cross sections, this trigger is expected to yield a $b \rightarrow e\bar{\nu}X$ purity of about 15%.

1000–2000 events from a Monte Carlo sample containing $b \rightarrow e\bar{\nu}X$ jets and on 4000–5000 Monte Carlo two jet (gluon–gluon) background event sample, where the jet p_t is > 15 GeV, and the pattern contains a seed tower with $E_t > 4$ GeV. The primary level calorimeter trigger thresholds implemented at CDF precludes reducing the seed threshold below 4–5 GeV for the neural net trigger, which is a secondary level trigger. Jets with p_t below 15 GeV do not produce enough energetic electron candidates, making the event generation inefficient. The events, produced in $\bar{p}p$ collision where the beam energy is 0.9 TeV, are generated using Isajet with CDF default fragmentation parameters [7] and passed through detector simulation and event reconstruction. New patterns not in the training sample are then used to estimate trigger efficiency after the net being optimized. Both the testing and training patterns must pass the following requirements in addition to CEM seed trigger tower $|\eta| < 1.0$: a) seed tower $4 < E_t < 15$ GeV, where the upper limit is to have the network improve its efficiency for identifying lower energy b's; b) a charged track with $p_t > 3.5$ GeV matching the seed tower in ϕ and η; and c) for Monte Carlo signal events, the seed tower shower must be from an electron from b decay. Fig. 4 shows the distribution of the output values of the trained neural net (the value of last node in fig. 1) for new test signal and background data. The trigger efficiency will depend on the exact threshold value we select in fig. 4 to separate signal from background. We choose a value of about 0.8 to reject more than 98% of the background. For Monte Carlo training samples, the network accepts $40.8 \pm 1.4\%$ of the $b \rightarrow e\bar{\nu}X$ and $1.4 \pm 0.2\%$ of the gluon patterns. For Monte Carlo testing samples, the values are $39.4 \pm 1.6\%$ and $1.3 \pm 0.2\%$, indicating that the separation performance generalized to new patterns. Appendix B provides further efficiency values when we train with varying amounts of patterns.

To check inaccuracies in the simulation, we use the data collected by CDF with loose trigger requirements to compare with our Monte Carlo expected efficiencies. We apply the same set of requirements, consisting mainly of stringent electron cuts, to both data and Monte Carlo to obtain a fairly pure sample of $b \rightarrow e\bar{\nu}X$ jets and then pass the pattern through the trained net. For our choice of network output threshold, for 5 GeV electron E_t, the efficiency for both real and simulated data is about 56%, and about 65% for 7 GeV electrons. The agreement between Monte Carlo and data for the signal is generally good and the neural net is not very sensitive to the deficiencies in the simulation. We use minimum bias events for a consistency check of our Monte Carlo estimate of background rejection. For events with seed $E_t > 4.0$ GeV and passing the same criteria as the neural net training sample, the network acceptance is $2.8 \equiv 0.8\%$ as compared to the $1.3 \pm 0.2\%$ from the Isajet gluon two jet events estimate.

Once the candidate electron p_t is greater than ~ 7 GeV the b purity of the sample is naturally high for an inclusive electron trigger. Then even simple, conventional electron triggers begin to do as well as a trained neural net trigger in signal-to-background. We find using minimal electron requirement data (the thresholds are looser than for the standard electron trigger) the network still does better at 7 GeV, providing a factor of two improvement in $b \rightarrow e\bar{\nu}X$ purity over that of the conventional CDF electron trigger for the particular neural net output threshold chosen.

Although we have not trained the neural net on a particular type of $b \rightarrow e\bar{\nu}X$ jets, the sample which passes the neural net requirements turn out to favor cases where the electrons are more isolated and the surrounding jet energy is lower than that of the typical $b \rightarrow e\bar{\nu}X$ jet. For example, the sum of the E_t in the eight electromagnetic calorimeter trigger towers adjacent to the seed tower for accepted patterns averages 1.7 GeV, but 4.1 GeV for rejected ones. The isolation is reduced if the required network output threshold loosened.

The relative values of the weights connecting the input layer and the first hidden layer provides some indication of the significance of a particular input variable. For our patterns, the maximum value amongst all of the weights after training is four times larger than the others and it belongs to the electromagnetic–hadronic seed tower, where the electron candidate is located. Therefore much of what the trained neural net recognizes is the electron. This is not too surprising because the typical pattern is an array of small energy depositions symmetric about a more obvious, larger deposition. This trigger is quite efficient on finding electrons from sources other than from b jets. A smaller 3×3 grid pattern would be more suitable for a neural net general electron trigger.

We make a preliminary comparison based on Isajet Monte Carlo b meson events (trigger seed $E_t > 5$ GeV) between the sample tagged with a neural net and the sample obtained via the conventional CDF electron trigger and analysis cuts [8]. For example, the average p_t of the electron candidate is 8.7 ± 0.1 GeV from neural nets, higher than 7.2 ± 0.1 GeV from conventional cuts. The average p_t of the parent b is however higher for the conventional $b \rightarrow e\bar{\nu}X$ events, 17.0 ± 0.2 vs 16.4 ± 0.2 GeV, but the peak p_t value is ~ 14.0 GeV for both samples. The angle between the electron and the b averages about 9.2° for both sets. The conventional CDF cuts to obtain a high purity $b \rightarrow e\bar{\nu}X$ sample include variables causing the electrons to be somewhat isolated, similar to the events accepted by the neural nets. The relative ratios of parent b type (i.e. $b_u : b_d : b_s$) and sibling D type or D* type species are the same for both samples. The average number of particles and the particle content of the decay products and of the underlying event also agree well. In summary, the kinematic distributions from neural net tagged $b \to e\bar{\nu}X$ are similar to conventionally tagged ones.

2.3. Trigger issues of neural nets

For trigger applications, the efficiency of the neural net must be fairly insensitive to the effects of electronic noise, dc offsets, and device variations. When we adjust the optimized weights by 1% (manufacturer specifications) and then reevaluate the trigger efficiency on the test data, the rates change by ~5%. Similarly, the trigger performance is affected by less than 5% if there is 40 MeV equivalent voltage offset in the circuit (e.g. in the stages involving operational amplifiers). To check the situation due to electronic noise, we add energy distributed Gaussianly about a mean of 100 MeV to each calorimeter trigger tower in the test patterns. As expected, this noise affects patterns containing low energy electron candidate more than those with higher energy. For seed tower E_t above 4 GeV, the rates for either noisy signal or background patterns change by about 20%; whereas, for E_t above 7 GeV, the difference is about 10%. This accounts only for the effect of positive voltage value noise. Unfortunately, if both negative and positive noise are included, our simulation of the trigger performance suffers a ~50% change in rates. We are training with patterns allowing for both positive and negative noise to try to reduce the sensitivity to this effect. Initially since positive voltages are used to represent calorimeter energy measurements at CDF and the neural net circuit is operated in the positive voltage region, we train the net with only positive input values.

The 80170NX unfortunately has a limited input signal voltage range of 0–3.5 V. We check the effect of rescaling our inputs to optimize running in this voltage region. Using the network weights actually optimized with the full scale data (1 GeV = 1 V), we run new patterns where the inputs are scaled down by a factor of 2 (2 GeV = 1 V). We find that the trigger efficiency remains about the same as before. We note however, that if we have to scale the inputs down further by factors of 5 or 20 and run these scaled-down patterns through the full scale trained net, then the result is very poor separation. If instead, we now reoptimize the net by starting with training data that is already scaled down, by 5, 10 or 20, we can achieve a separation efficiency comparable to before by selecting an output threshold requirement different from the previous 0.8 value. However, the distributions of the network output for new patterns are somewhat different than shown in fig. 4, with the peak for background data shifted closer to zero and signal data forming a broader peak. The result is that the new cut ends up on the peak or on the higher end tail of the distribution for signal patterns. It turns out this network output distribution resembles that of a 1×1 training pattern case where we fed the network only the seed hadronic and electromagnetic E_t values. The scaling down causes the 5×5 pattern to look like only a single electron candidate tower (seed) with energy, and the rest of the tower energies be comparable to values due to noise. Such a scaled-down 5×5 pattern kinematically resembles that of the 1×1. For both the 1×1 and the scaled-down 5×5 case, the minimization results in events containing higher seed E_t's to have a higher probability of being identified as signal.

3. Second application of neural net technology

We now describe an altogether different way of using analog neural net chips – to carry out conventional, nontrained trigger algorithms. The basic architecture of the 80170NX chip permits using it to do arithmetic sums, multiplication, fixed ratio checks, and logical ANDs and ORs. For the neural net $b \to e\bar{\nu}X$ trigger we use a 5×5 calorimeter grid centered about a seed electromagnetic (e.m.) tower with a large energy deposition (electron candidate). When there are no tracks associated with the seed e.m. tower, then the energy deposition is very likely a photon candidate; isolated photons are of particular interest because they probe QCD and are signatures for new physics. We may chose one of the following procedure to implement isolation criteria. The first is an example of using the chip to perform sums and threshold requirements, and the second to check whether ratios pass a certain value (e.g., $A/b < 10$).

1) Require the sum of the energy deposited in the 41 calorimeter towers, adjacent to and behind the seed tower (two layers, 5×5 minus the corners), to be small and less than some value α. This is done by setting the weights for the 41 towers in the summation in the sigmoid function to -1.0, and 0 for all other towers. The offset parameter b is set to α:

$$\text{neural net output} = \frac{1}{1 + e^{-(\sum_{i=1}^{41} x_i + \alpha)}}.$$

When the value of the output is less than 0.5, then the isolation requirement is satisfied, otherwise not.

2) Alternatively, the energy deposited in the 41 towers may be required to be only some fraction of that deposited in the seed tower, for example, $\frac{1}{10}$. Now, the weights for the 41 towers should be set to -10,

and 1 for the seed tower, and zero for the bias parameter b:

$$\text{neural net output} = \frac{1}{1 + e^{-(-\sum_{i=1}^{41} 10x_i - 1x_{\text{seed}})}}.$$

Again the output threshold is set to be 0.5.

The neural net chip has several good features which compensate for this awkward way of performing arithmetic. First, our particular chip is very compact (45 × 45 mm^2), but it allows 64 parallel inputs and contains 64 hidden nodes, permitting 64 individual operations to be carried out simultaneously. Although it is possible to use conventional ICs like operational amplifiers to do the same arithmetic, the number of circuit components required would be very much larger. Second, since we are designing circuit boards for the $b \to e\bar{\nu}X$ trigger, we find it convenient and economical that the exact same design can also handle conventional trigger functions. The third and main reason is the ability to reprogram the chip's weights and threshold values via EEPROM technology to accomodate trigger changes. We can design physics trigger circuits around this device with almost no algorithm in mind a priori, and then afterwards select a particular algorithm. Input voltage ranges may be rescaled, and offsets may be removed by changing the weights and biases. It is important to note for this analog device that one can revise even the type of arithmetic operation desired for the algorithm, such as from addition to a ratio-check as shown in the above example for photons.

Depending on application, the 80170NX need to be improved in speed (~ 3 μs) and in the voltage range of the input and output signals (between 0–3.5 V) for both the trained and untrained use of the chip. For untrained network applications, it is possible to use simpler functions instead of a sigmoid for the nodes, so long as the $\Sigma_i w_i x_i + b$ structure is retained. This can speed signal processing time, reducing from 3 to 0.8 μs, values based on the characteristics of our chip. The complexity of the 80170NX should be reduced as well which should lead to improved performance and flexibility. The chip was designed to allow a two layer network by clocking results of the previous layer through the chip again. Chips containing only a single layer network are easier to use and multiple chips may be cascaded for multilayer network applications.

4. Conclusion

We have considered the multilayer feed-forward type of neural net to recognize $b \to e\bar{\nu}X$ jets at a $\bar{p}p$ collider. These nets must first be optimized with a large data sample, requiring considerable computing resources. Afterwards we load the weights into the 80170NX to operate it in the CDF trigger. We have provided a primer on training the network in the appendix and have described some shortcomings of utilizing trained neural nets. We will also use the analog chip to perform conventional trigger tasks in parallel with the trained neural net. Both the $b \to e\bar{\nu}X$ and conventional applications will have trigger cross-checks to ensure a full understanding of these new techniques. The technology has great potential because it can distinguish complex patterns and it allows changes to trigger algorithms in a way never permitted before. We will determine if neural nets can expand our choice of techniques for fast triggering which thus far is limited because sometimes conventional cuts may be powerful but are difficult to implement electronically. We look forward to the 1992 data run.

Acknowledgements

We thank CDF for use of the Monte Carlo and event reconstruction code. We appreciate useful comments from K. Riles and R. Thun. We benefited from neural net studies by N. Chriss, B. Denby, and particularly Zhaoping Li. This work was supported by the Department of Energy (DOE-DE-AC02-76ER011112), the Sloan Foundation, and the Texas National Research Laboratory Commission.

Appendix A

Neural net training algorithm

The type of neural net we use is known as the feed-forward, multilayer perceptron configuration which accepts patterns with either continuous valued (e.g., particle energies) or binary inputs. The theory of neural nets is discussed extensively in refs. [6,9]. We consider here instead the practical aspects of the training procedure. The configuration is widely applicable and basic to some other networks.

We use the algorithm of ref. [6] for obtaining the optimal values for the parameters, weights w and biases b, contained in our neural net function S:

$$S_l^k \equiv g_l\left(\sum_j w'_{jl} f_j\left(\sum_i w_{ij} x_i + b_j + b'_l \right)\right), \tag{A.1}$$

where i indexes the 50 inputs, j the 10 hidden nodes, k the pattern, and l the 1 or 2 output nodes. S defines the neural net configuration (fig. 1), which for us consists of a layer of 50 inputs x_i (the zeroth layer), a hidden layer f_j with 10 nodes (the first layer), and an output layer g_l usually with one node (the second layer). The summation is over the number of nodes feeding into each network layer. All f_j and g_l have the same functional form, called a sigmoid function (represented by the rectangles in fig. 1):

$$f_j = \frac{1}{1 + e^{-(\sum_i w_{ij} x_i + b_j)}}. \quad \text{(A.2)}$$

The goal is to separate the training sample into two classes via an appropriate S, e.g. to obtain a desired response value ($S^k_{l(\text{target})}$ for the kth pattern) of S to be ~ 0.9 for input signal events and ~ 0.1 for input background events. The optimum set of weights and biases is found by a least squares minimization of the differences E between the actual responses $S^k_{l(\text{actual})}$ and the target responses. The weight values for the minimum point are reached through successive approximations; in each iteration, the weights are changed by an amount proportional to the partial derivatives of the function E with respect to the w value, and similarly for the biases. One possible choice for the function E is:

$$E \equiv \tfrac{1}{2} \sum_l \sum_k \left(S^k_{l(\text{target})} - S^k_{l(\text{actual})} \right)^2, \quad \text{(A.3)}$$

where k indexes a summation over the number of training patterns, and l over the number of output nodes (for cross-check purposes, we sometimes use two output nodes). In short, the training procedure samples the function E in weight space until yielding an optimized function S in input-variable space to separate signal from background. In practice, the procedure consists of the following:

1) The net is started in an arbitrary state by generating different small random values for the w_{ij} and b_j. For a reasonable set of starting weights, the final result does not depend on the initial choice. During the first pass of the inputs through the neural net, we check that the response value of S for at least most of the patterns is within the 0.1–0.9 region; otherwise we rescale the weights (see appendix D on the scale of training parameters). We use initial weight and bias values between -1 and 1, but for the weights there is a multiplicative factor of $(\bar{x}_i/N$, where $\bar{x}_i$ is the average value of the inputs to a particular layer and N is the total number of inputs to that same layer).

2) The value of f_j for each node j is calculated using the current values of the w_{ij} and b_j. This is referred to as presenting the input data to the net. Each input pattern should have a fixed desired response, $S^k_{l(\text{target})}$, associated with it. During training, we have set the target output value to 0.9 for patterns from signal events, and 0.1 for background.

3) The weights are adjusted at iteration $(t+1)$ based on the previous state of the neural net at iteration t

$$w_{ij}(t+1) = w_{ij}(t) + \eta \, \Delta w_{ij}(t) + \alpha\left(w_{ij}(t) - w_{ij}(t-1)\right), \quad \text{(A.4)}$$

where i indexes the nodes (or inputs) in a previous layer and j, the nodes in the next layer. The parameters η and α, scale factors input by hand, are described in appendix D. For weights connecting each of the hidden nodes i and the output node j

$$\Delta w_{ij} = -\sum_k \delta^k_j f^k_i, \quad \text{(A.5)}$$

where k indexes a summation over the total number of patterns and

$$\delta^k_j = \left(f^k_j\right)\left(f^k_j - S^k_{l(\text{target})}\right)\left(1 - f^k_j\right) \quad \text{(A.6)}$$

(note: for the jth output node, f^k_j is the same as the actual final response of the net, S^k_j, from eq. (A.1)). For the weights connecting the inputs x_i and the jth node in the hidden layer, the term Δw_{ij} is:

$$\Delta w_{ij} = -\sum_k \left(\left(f^k_j\right)\left(1 - f^k_j\right) \sum_l \delta^k_l w^k_{jl} \right) x^k_i, \quad \text{(A.7)}$$

where l is over all nodes in the next higher layer (output) and δ^k_l is the δ^k_j given above. The biases b_j are updated in the same way as the weights.

Steps 2) and 3) are repeated until E stops fluctuating and an arbitrarily small value for E is reached. The procedure is slow but fairly stable and is intended for a priori training and then later use in a system with predetermined weights. For an electronic implementation, the fact that the weight changes (derivative of the sigmoid function) depend only on the f_j values and facilitates placing the actual IC chip in the training loop.

Typically as suggested in ref. [6], the neural net weights are updated after pairs of signal and background patterns are presented to the net. Instead, we present all of the patterns to the net prior to each weight update (the summation over k). By doing so, we have averaged over the effect of all of the input events every time. Averaging requires scaling η down. For example, we use η values ranging approximately from 10^{-5} to 10^{-4} when a few thousand patterns are presented prior to each weight update; whereas, when only pairs of patterns are presented, η values from 0.1 to 0.5 are more appropriate. This averaging method allows more flexibility in that the order of the input pattern does not matter, and we can input more patterns from one class than from the other. For a $b \to e\bar{\nu}X$ trigger at a $\bar{p}p$ collider, we expect many more background than signal events and thus to more closely span the larger variety of background patterns, we use about 3–4 times more background than signal events.

Appendix B

Training sample

This appendix and the next two address how we select our net configuration, the significance of some of the training parameters and their values. Generally, the most important consideration is the training data themselves. A neural net cannot separate patterns which are intrinsically inseparable.

The intrinsic overlap between signal and background patterns places an upper limit on the network's ability to separate the two. If one knows something about the variables characterizing the events beforehand, or has other means of enhancing the signal-to-background (e.g. stiff track in addition to a neural net cut), one should limit the diversity of the patterns and the range of the input variable values of the patterns to train on only a subset of the data to improve the separation efficiency. As a simple example, we know that $b \rightarrow e\bar{\nu}X$ jets are more similar to gluon jets when the E_t of the electron is lower and thus expect a network to do poorer on lower energy patterns. With our signal $b \rightarrow e\bar{\nu}X$ patterns and our background gluon jets, we started with training patterns where the seed electromagnetic (CEM) tower energy ranged from a) 3 to 15 GeV, then narrowed it to b) 15 GeV, and finally restricted the sample further by c) requiring a stiff track to be associated with the candidate electron shower. Because of the large background cross section, we must set the neural net trigger thresholds to accept at most about 1% background. For approximately the same amount of training data, network topology, and η and α parameters, and after the network performance efficiency has generalized, we find for $\sim 1\%$ background, the neural net efficiency on new signal patterns, passing the same requirements as the training data, improved from a) $\sim 8\%$, to b) $\sim 14\%$, and finally to c) $\sim 30\%$.

Even after we narrow the kinematic characteristics of the training sample, there may still be variations, and the optimized neural net will generally be efficient in distinguishing only a subset of the patterns. Sometimes, the trained-net's response distribution for, say, the signal data, will have several peaks, reflecting the existence of subsets with different kinematic properties within the sample. The distribution may be used to guide the pruning of the training patterns, to try to steer the neural net to recognize a particular data subset.

The number of patterns and the relative amount from each class (signal and background) affect the final separation. Table 1 shows different cases for patterns from a 15 GeV jet p_t with a 4 GeV E_t electron candidate generated from the Isajet Monte Carlo. To compare the cases we check the neural net acceptance efficiency at an output threshold value that allows the network to accept at most $\sim 1\%$ background (last column of table 1). The 1% value is a level permitted at CDF based on luminosity, cross sections and dead time considerations. The relative low efficiency of 20% (first row of table 1) made it apparent that more than ~ 1500 background patterns are needed for training. With too few patterns, the efficiency is not only lower but is also more likely to fluctuate (by a much as 40%) at successive efficiency evaluation intervals; whereas, the fluctuation is typically only 5% by using > 3000 background training patterns. In addition, the net does not generalize so well: the discrepancy between the acceptance of background training and new test patterns widens by a factor > 3, as the number of weight updating iterations is increased. To investigate whether further increasing the total number of patterns or the ratio of background patterns compared to signal is advantageous, we loosen the output threshold cut to improve the statistics of the background accepted (signal detection efficiency of 50%, fourth column of table 1). There is no significant improvement in doing either. Because it requires much CPU to generate more training data, by assuming that 4224 events adequately span the set of background patterns, we have checked that a yet larger ratio of background-to-signal patterns shows no improvement by multiplying the effective weight change by ten for background events. Aside from generation time, the CPU needed for training also increases with the number of patterns: for a total of 7000

Table 1
The effect of varying the number of training patterns and the relative amounts of signal-to-background: the fourth column shows the acceptance of new background patterns when the trained net passes new signal patterns with 50% efficiency; and the last column gives the signal efficiency when the background acceptance is required to be no more than $\sim 1\%$

Number of training patterns		Background-to-signal ratio	Background acceptance (50% signal)	Signal acceptance (1% background)
Background	Signal			
1452	1213	1.2	$3.0 \pm 0.2\%$	$\sim 20\%$
3109	1213	2.6	$2.9 \pm 0.2\%$	$\sim 30\%$
4224	1213	3.5	$2.2 \pm 0.2\%$	$\sim 30\%$
42240	1213	34.8	$2.6 \pm 0.4\%$	$\sim 25\%$
5122	1777	2.9	$2.8 \pm 0.2\%$	$\sim 30\%$
3109	2182	1.4	$2.8 \pm 0.4\%$	$\sim 30\%$
4224	2182	1.9	$2.7 \pm 0.4\%$	$\sim 30\%$

patterns, 50 inputs per pattern, and one layer of 10 hidden units, it takes about 8 hours on a 7–10 MIP machine; whereas, 2500 patterns take about 3 h. Based on these considerations we typically now use 4000–5000 gluon background and about 1200 $b \to e\bar{\nu}X$ signal patterns for training.

Appendix C

Network configuration

The topology of the net is important because it helps determine the region of space containing the minima points. Larger nets, with more hidden units and associated weights, can parameterize more characteristics of the training patterns. However, more free parameters also require much more data and processing time to optimize. As with any parameter fitting procedure too little data force the parameters to take on a very specific value and "memorize" the input training set, in which case the network performance does not generalize to new test patterns. For the multilayer perceptron type of network, the number of inputs, the number of layers, the number of nodes within each layer, and the number of weight connections between nodes define the topology.

The numbers of layers determines what shape the final surface separating the two classes of events may take [9]. Three layers, in addition to the input layer, where the nodes are continuous functions, can provide the most general shape. Single layer networks involve only linear combinations of the input variables and are adequate only when the signal and background occupy different sides of a plane in the input variable space. Two layers allow a region of space to be approximated by either an open or closed convex surface. We expect the energy variables for signal and background jets each to occupy a continuous region of space where the surface that best divides them is not necessarily a plane; so, we use only two layers: one hidden layer and one output. Furthermore, extra layers result in an electronically slower trigger.

We now examine the number of nodes used in each of layers, which also then determine the number of weights (interconnections). For the input layer, we have 50 units (5×5 grid), a choice determined by the size of the jets. A 6×6 grid yields about the same separation efficiency but requires more electronics channels to implement and processing time to train, and a smaller grid is insufficient to contain the typical jet. We use 10 nodes in the hidden layer and one in the output. This results in a total of 510 weights plus 11 biases, 521 parameters to be optimized! Clearly, the amount of training data must be large. However, not all of the weights matter; the larger weight values are usually of order 1.0 and among the 500 weights between the input and hidden layer, about half of them are smaller than 0.05. We also try starting with fewer nodes in the hidden layer prior to training, resulting in fewer small-valued weights after training. For instance, for one hidden layer with two nodes, only 12 of the 100 weights connecting the input and the next layer, are less than 0.05. The range in the values of the weights, the achievable separation efficiency and purity by an appropriate choice of neural net output threshold are similar to the 10 node case. However, the response distribution for signal patterns with two hidden units contains two prominent peaks, one centered around 0.14 and the other at 0.8, unlike the ten hidden unit case shown in fig. 4 which has one prominent peak. The signal events in the two peaks differ most noticeably in the degree of isolation of the electron. The ten hidden unit case favors patterns with isolated electrons in a much more gradual way, producing only one peak. The background distribution for both cases has only one peak, around 0.13. As the number of hidden nodes is increased from 2 to 10, the signal events in the first peak (at 0.14) diminish and migrate to the second peak. The first peak does not continue to shrink as we go to 15 hidden units for our fixed training sample size.

Appendix D

Training parameters

After defining the network topology and thus the space containing the minima and the total number of variables to optimize, the scale parameters α and η in eq. (A.4) then determine which of the points of the function E the network can possibly reach ("step to") during training. The variable η is a "step size" to move the neural net from one point to the next along the direction of the slope of E towards a minimum. The value of η should be chosen to be small enough so that the minimization samples the points in a fairly smooth fashion. The α term is an optional variable; it further smooths the weight changes since it brings in a term involving the previous weight update. If the scale of these parameters takes S to an extreme end of the sigmoid function, 0.0 or 1.0, the training tends to collapse (the value of E drops to and remains zero suddenly) or to be slow. The weight changes are proportional to the derivative of the sigmoid function, but at the extreme ends, the derivative is very small or zero. To sample as many points as possible, η may be set to smaller values and the training repeated with different initial weights. We prefer smaller values for η also because the network output distribution seems to be smoother. We notice that for different choices of η, the resulting shape of the network output distribution might vary. Perhaps, this indicates that the convergence is not final or that there are different minima. (However, the separation efficiency is similar for the different cases with varying distributions.) The training sets with higher η values appear to classify our data (several peaks in the distribution) rather than perform a single separation (two peaks, signal vs background). Small changes to the other variable, α, affect the training much less than changes to η [6]. We try values ranging from 0.0 to 0.8, but usually used 0.3 or 0.5 for α.

For each training pattern, we associate a desired scalar response S^k_{target} in eq. (A.1). We use 0.1 for background and 0.9 for signal. Choosing values like 0.05 and 0.95 would presumably create a greater distance of separation; but, for our particular setup, using target values closer to 0.0 and 1.0 does not improve the final results.

One more consideration in training is the number of weight updating iterations, a value which we determine empirically from two main factors: the value and stability of E (eq. (A.3)), and a comparison of the separation efficiency for the new test vs the training data. We require the acceptance efficiency to generalize to the test data. (This however does depend on how many training patterns we use. Without enough data to span the diversity of the entire pattern space, the network acceptance on training and new test data begin to diverge after a certain number of weight updates – though this is not true with larger samples of events.) The performance efficiency improves from iteration to iteration, but slowly so that it is adequate to check the performance after every 200 updates. We run about 3000 iterations for each training set, but for our patterns, 1000 iterations are usually sufficient to reach the best separation. Further weight updates do not improve the efficiency. In practice, we sometimes use two sigmoid units in the output layer connecting to each one of the hidden units, although only one is shown in fig. 1. Training signal patterns are required to be 0.9 from output unit one and 0.1 from the other, and exactly the opposite for background patterns. Once the weight updating has reached optimum, the two sets of weights connecting the hidden and each of the two output nodes become antisymmetric (the value of one set is the negative of the other, a likely result of the fact that the sigmoid function equals one minus the sigmoid with the exponent term negated). Reaching this antisymmetry helps confirm when to stop training. We find that values of the weights between the input and hidden layers grown but do not typically display any obvious symmetry even after many iterations, de-

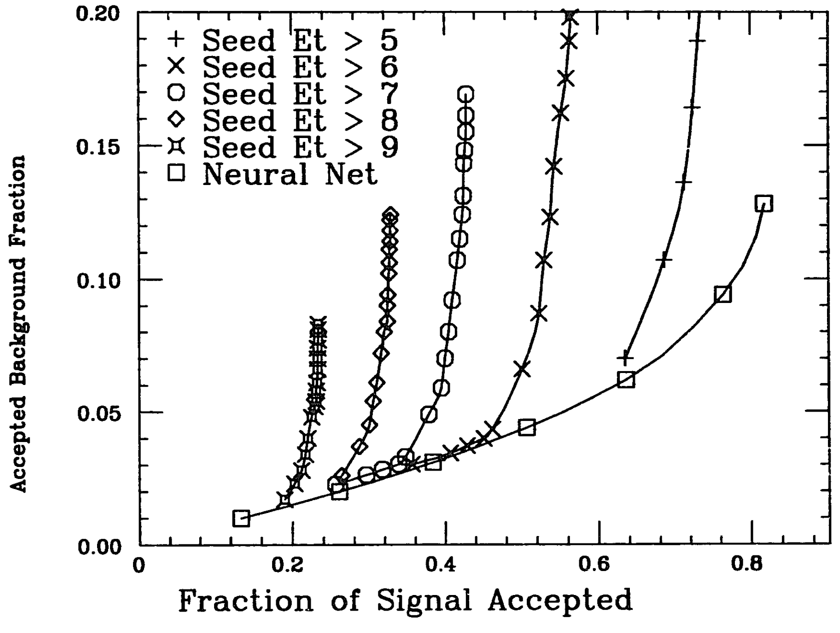

Fig. 5. A comparison of neural net and threshold cut acceptance for signal and background patterns with only two inputs. The conventional selection curves correspond to different had/e.m. ratios after requiring the electromagnetic seed E_t to be above some threshold. The neural net curve gives the acceptance for different network output values. Though shown for only two cases, all of the had/e.m. curves only asymptotically approach the network curve, but never cross. A neural net performs as well as or better than conventional selection requirements, the relative improvement increasing with increasing variables and correlations among variables.

spite the symmetry of the 5×5 pattern itself; they do not give much of a clue to the number of necessary iterations.

Appendix E

Assessment of our neural net

Mathematical considerations of neural net algorithms are still under study [6,9]. There are possible problems at each stage of the algorithm we use. For fairly complicated networks and jet patterns like ours, it is not possible to check in advance whether a solution (convergence of the minimization) exists or whether there are many local minima. It takes experimentation to determine the best region for the training parameters η and α. Afterwards, we make many training runs using different parameters in that region to sample as much of the space containing the minima as possible. However, the minimization can collapse which occurred 5–10% of the time for us even after we set the order of magnitude of the training parameters in a reasonable region.

There are other drawbacks in using the algorithm. The optimal techniques for our particular set of patterns may not apply to another application. Also, the best separation is not necessarily the most beneficial; it may not yield accepted patterns with the most desirable kinematic characteristics. It is difficult to control the exact nature of the final sample of accepted events. The saving feature of the algorithm and the network is that in many cases studied, a desirable minimum is reached and signal and background patterns are indeed separated, as for the $b \rightarrow e\bar{\nu}X$.

Our primary goal has been to achieve the best separation rather than to obtain a sample of patterns with a particular set of characteristics. All of our good training runs result in approximately the same final separation efficiency, which suggests we have reached a limit. Testing the efficiency and plotting the pattern characteristics every couple hundred of iterations is one way of determining whether desirable separation has been reached. The lack of fluctuations in E and in the network acceptance efficiency after about 500 weight updates make it unlikely for us to accidentally skip over the "best" minimization point even though we do not check the efficiency after each update.

Because there is an upper limit for separating two sets of patterns, it is interesting to verify on a simple example that neural net triggers are better able to reach this limit than conventional one-dimensional threshold requirements. We train on a 1×1 pattern consisting of two inputs, one electromagnetic and one hadronic tower E_t. With only two inputs, there is not much choice in the way of cuts, other than to demand each E_t or the ratio of the two E_t (had/e.m.) exceed some value. In this example, there are few enough variables that we can optimize the signal-to-background ratio by inspection, and we find that requirements on electromagnetic E_t and had/e.m. to be best. Shown in fig. 5, is the acceptance curve for different values of had/e.m. for E_t above some fixed threshold; also shown in the neural net's acceptance for different choices of the output value. The neural net does indeed to better than conventional cuts, but with so few variables, the improvement is only slight. For multidimensional variables, however, a neural net becomes more effective and practical for achieving the best separation, particularly when the signal is hard to distinguish from background. Knowing this, we can also use neural nets in another way. They can help determine the best range of values to set one-dimensional cuts for the different variables used in a conventional trigger or analysis, or to show whether a particular set of variables has any hope of extracting a signal.

References

[1] M. Holler, S. Tam, H. Castro and R. Benson, (Intel Inc.), Proc. Int. Joint Conf. on Neural Networks, June 1989, Washington, DC Vol. II, p. 191.

[2] N. Ellis and A. Kernan, Phys. Rep. C195 (1990) 23.

[3] L. Balka et al., Nucl. Instr. and Meth. A267 (1988) 272; and S. Bertolucci et al., Nucl. Instr. and Meth. A267 (1988) 301.

[4] D. Amidei et al., Nucl. Instr. and Meth. A269 (1988) 51.

[5] B. Denby et al., IEEE Trans. Nucl. Sci. NS-37 (1990) 248.

[6] D.E. Rumelhart and J.L. McClelland, Parallel Distributed Processing Explorations in the Microstructure of Cognition, vol. 1 1988, MIT Press; and
R.P. Lippmann, IEEE ASSP Mag. April (1987) 4.

[7] F. Paige and S. Protopopescu, Brookhaven National Lab Report, BNL 38034, 1986; and
F. Abe et al., Phys. Rev. D43 (1991) 664.

[8] A.R. Baden, representing CDF collaboration at the SLAC Summer Institute 1990, Fermilab Conf. 90-255-E-Rev.

[9] J. Hertz, A. Krogh and R. Palmer, Introduction to the Theory of Neural Computation (Addison-Wesley, 1991).