Report *SYCON-88-02*

SOME REMARKS ON THE BACKPROPOGATION ALGORITHM FOR NEURAL NET LEARNING

Eduardo D. Sontag
Department of Mathematics
Rutgers University
New Brunswick, NJ 08903
(201)932-3072
sontag@fermat.rutgers.edu

ABSTRACT

This report contains some remarks about the backpropagation method for neural net learning. We concentrate in particular in the study of local minima of error functions and the growth of weights during learning.

Rutgers Center for Systems and Control, 1988

1 Introduction

Backpropagation is probably the most popular method currently being used for neural net learning. It was introduced in the neural literature by Rumelhart and Hinton in the seminal work [PDP]. See for instance [Hi87] for an introduction and references to current work. It can be understood as an iterative gradient technique for nonlinear least squares fitting, the cost function corresponding to a clustering problem to be solved. Its study gives rise to many open mathematical problems. This note makes some remarks about factors affecting the ultimate performance of backprop and its variants.

A basic issue that must be understood when applying a gradient technique is that of the structure of the set of local minima of the error function. (Under "local minima" one must include limiting values when some of the weights tend to infinity, of course.) We believe that the local minima structure may be fairly complicated, and we propose a rigorous study of this issue, which has puzzled many practitioners. The state of knowledge about this problem is rather limited. The following quote, from the excellent book [Ar86], is typical (the emphasis is ours):

> "Although *no formal proof is available*, simulation shows that the scheme avoids many of the false minima that bedeviled other methods. Rumelhart et al. found the algorithm to be rather robust when tested on several hundred problems. In a *few cases, local minima* were found, but they were *easy to* escape from. If there are enough units, the method can attain $E = 0$; if not, the weights can be trained to minimize E, a result that is optimal in the sense of least squares."

Just as Arbib, we do not know of any serious analysis of the problem. However, we feel on the basis of our experience that local minima are much more ubiquitous than usually thought. Further, it is not clear why they should be easy to escape from. It is possible that the particular practical problems in which the method has been tested exhibit some structure which makes them better behaved. If this is the case, it is by far one of the most important issues to be understood regarding backpropagation. Conversely, if local minima do tend to appear, as our general mathematical analysis suggests, then this will have to be taken into account in learning algorithms. For instance, it is then reasonable to include stochastic relaxation techniques such as simulated annealing (see e.g. [KGV83], [SS85])..

To understand the problem, we shall look here at the simplest possible version of backprop, that of networks with no hidden neurons and just one output neuron, with m input neurons. This is the classical *perceptron*, for which the full power of backprop is not really needed. Since we wish mainly to illustrate pathologies, we prefer to start with this simple case. It may be reasonably be expected that those difficulties that arise here will of course also show up in more complicated networks. Further, it seems to be the case experimentally that in the general case, quoting [KH88], section 2.2:

"When the weight values are close to the desired values, the lower layer weights are changing very slowly and can be assumed to be almost fixed..."

That is, the problem then becomes one of learning with no hidden units just as considered here.

2 The minimization problem

Mathematically the input/output behavior of the systems considered here can be described as follows. Given a vector input

$$u = (u_1, \ldots, u_m)' \in \mathbb{R}^m,$$

(prime indicates transpose) the corresponding output is

$$y = \theta(\sum_{j=1}^{m} u_j x_j) \in \mathbb{R},$$

where the x_i's are certain fixed real numbers, the *weights* associated to the different input channels. The function θ is of sigmoid-type, in the sense to be described later.

This is a purely static situation, in contrast to other techniques such as Boltzmann machines or Hopfield nets in which outputs are only considered to be obtained after a certain amount of settling of the system. In these other situations, there is typically feedback between input, output, and/or intermediate neurons, and only equilibria solutions of the corresponding dynamical systems are of interest.

A typical "learning" problem is specified by providing a set of desired input/output pairs

$$(u^1, y^1), \ldots, (u^n, y^n), \qquad (*)$$

the goal being that of finding weights x_i's such that the total square error

$$E = \sum_{i=1}^{n} \left[\theta(\sum_{j=1}^{m} u_j^i x_j) - y^i \right]^2 \qquad (**)$$

is minimized, where u_j^i denotes the j-th component of the vector u^i. The backpropagation algorithm is an iterative gradient method for minimizing this error. Typically the integer m, the number of weights to be found, is much smaller than n, the total number of learning runs.

In the case treated here, in which there are just 2 layers, the *perceptron learning theorem* insures that any linearly separable relation can be learned in finite time thorugh the classical perceptron learning procedure. However, the problem of minimizing the above

square error for arbitrary input/output pairs, via a gradient system, is mathematically entirely different.

Usually, the output values y^i are binary, corresponding to classification problems. The ultimate objective is in that case that of obtaining a neural net –i.e., a function of the type $f(x) = \theta(\sum_{j=1}^m u_j x_j)$,– which classifies the given vectors $u^1, \ldots, u^n$ as belonging to one of two classes ("examples" and "counterexamples"). One then introduces the corresponding pairs (u^i, y^i) with $y^i = +1$ if u^i is in the first class, and $y^i = -1$ otherwise.

As a starting point, we wish to show the falsity of the "folk fact", often repeated, that there is in the case of no hidden neurons a unique local minimum to which a gradient procedure will globally converge. This is far from true, even in the most simple case in which there is only one neuron, one input, and two experiments (in the above notation, $n = 2$, $m = 1$). In order to present our remarks we first recast the problem in slightly different terms. This will allow us to see the problem in a form which is mathematically more convenient.

Given a set of input/output pairs (*), we let A denote the $n \times m$ matrix whose (i, j)-th entry is u^i_j, and y the vector whose coordinates are the y_i's. We assume that the entries of y are all equal to $+1$ or -1. (Alternatively one could use $\{0, 1\}$-valued y, but the analysis is somewhat easier with our convention.) The (fixed) function θ is assumed to be a smooth strictly increasing and odd mapping

$$\theta : \mathbb{R} \to (-1, 1)$$

with

$$\lim_{z \to \infty} \theta(z) = 1, \ \lim_{z \to -\infty} \theta(z) = -1,$$

such as $\tanh(\cdot)$ or $(2/\pi)\tan^{-1}(\cdot)$. Finally, we let T be the mapping corresponding to applying θ to each coordinate,

$$T : \mathbb{R}^n \to \mathbb{R}^n : z \mapsto (\theta(z_1), \ldots, \theta(z_n))'.$$

We wish to understand the behavior of the gradient system corresponding to

$$\|y - TAx\|^2,$$

seen as a function of $x \in \mathbb{R}^m$. The first step is then that of studying the local minima of this function, including the limiting values where some coordinates of x become unbounded. The latter values correspond, when using a gradient method, to diverging solutions. (In practice, learning algorithms typically impose a priori bounds on possible weights, and diverging solutions are stopped when these bounds are achieved. We shall have some remarks about this later.)

Let $\mathcal{V}$ be the image of the linear operator A. Since T is a diffeomorphism of $\mathbb{R}^n$ with the open subset $(-1, 1)^m \subseteq \mathbb{R}^n$, $T\mathcal{V}$ is an embedded submanifold of $\mathbb{R}^n$. So the problem is closely related to that of studying the points that minimize the distance from the given point y to the closure of $T\mathcal{V}$,

$$\operatorname{dist}(y, \operatorname{clos} T\mathcal{V})^2.$$

Once that such minimizing points are found, it is a linear algebra problem for finite points to obtain the corresponding x's. If A has rank less than m, there will be an affine subspace of such x's corresponding to each minimizing point; this "subspace singularity" is observed in experiments with backpropagation but is totally uninteresting from a theoretical viewpoint. The real interest here is in the points in $\operatorname{clos} T\mathcal{V}$. (In the classical least squares problem, which would correspond to having no mapping θ, the pseudoinverse solution then picks the x of minimum norm that maps into a minimizing point.)

Since θ was assumed to be odd, the problem can be reduced to the one where y is the special vector

$$e = (1, \ldots, 1)'.$$

If $y \neq e$, it is only necessary to change the signs of suitable entries of the matrix A (equivalently, modify $\mathcal{V}$,) to reduce to this case. Thus, our problem is finally that of minimizing (locally and globally)

$$\operatorname{dist}(e, \operatorname{clos} T\mathcal{V})$$

where $\mathcal{V}$ is an m-dimensional subspace of $\mathbb{R}^n$. Note that $\operatorname{clos} T\mathcal{V}$ is compact, being a closed subset of $[-1, 1]^n$, so there is at least one global minimum. The first interesting case is that of $n = 2, m = 1$. We now show that already in this case, more than one local minimum may exist. The subspace $\mathcal{V}$ may be classified into one of three types:

1. the span of a vector of the form $\begin{pmatrix} 0 \\ a \end{pmatrix}$,
2. the span of a vector $\begin{pmatrix} 1 \\ a \end{pmatrix}$ with $a \geq 0$, or
3. the span of a vector $\begin{pmatrix} 1 \\ -a \end{pmatrix}$ with $a > 0$.

In the first case

$$\operatorname{clos} T\mathcal{V} = \{0\} \times [-1, 1],$$

so the distance is 1 and is achieved at the unique point $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$, and there are no other local minima. In the second case, if $a = 0$ there is again a unique minimizing point, $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$, while if $a \neq 0$ the point e is itself in $\operatorname{clos} T\mathcal{V}$, since

$$\lim_{z \to \infty} \begin{pmatrix} \theta(z) \\ \theta(za) \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix},$$

so the distance is 0 and is achieved only at e. There are no local minima: parameterizing the squared distance function as

$$E(x) = (\theta(x) - 1)^2 + (\theta(ax) - 1)^2$$

we have that its derivative

$$E'(x) = 2(\theta(x) - 1)\theta'(x) + 2(\theta(ax) - 1)^2\theta'(ax)a$$

is always negative because the first term is negative and the second is nonpositive.

The last case is much more interesting. We shall show that for a subspace $\mathcal{V}$ spanned by the vector of the form $\begin{pmatrix} 1 \\ -a \end{pmatrix}$ with large a, at least two local minima may appear. Actually, we show that in this case there is a local maximum for E as a function of x (with finite x), from which the conclusion will follow. Since both

$$\begin{pmatrix} 1 \\ -1 \end{pmatrix} \text{ and } \begin{pmatrix} -1 \\ 1 \end{pmatrix}$$

are in $\operatorname{clos} T\mathcal{V}$, the minimum distance is at most 2. Note also that

$$\operatorname{dist}(e, T\begin{pmatrix} 1 \\ -a \end{pmatrix}) > 2$$

for large a. Indeed, the square of this distance is

$$(\theta(1) - 1)^2 + (\theta(a) + 1)^2;$$

as $a \to +\infty$ the second term approaches 4 while the first is a positive constant.

Thus for $x = 1$ the cost in (**) is larger than 4, while as $x \to \pm\infty$ this cost approaches 4. It follows that there is a finite maximum and that there is at least one local minimum (possibly infinite) on each side of $x = 1$. In fact, further analysis shows that there is one local (actually, global,) minimum in the interior (x finite) and one at $x = +\infty$.

Figures 1 and 2 show respectively a typical graph of the squared distance function E, for cases 2 and 3 (with large $-a$). These graphs can be understood better when seen in conjunction with the plots of the images $T\mathcal{V}$ for those two cases, figures 3 and 4 respectively. (The function $\theta = \tanh$ was used here.)

3 Diverging weights

Experimenters with backprop typically make the algorithm stop learning when performance in new problems is adequate, for instance, when correct classification is attained for a set of consecutive inputs. It is our opinion that this stopping criterion may not be adequate if the ultimate objective is to generalize from examples. For such an objective, it may be important to force weights (the "x's" in the previous section) approach their optimal values. Looking at networks that have "learnt" a certain classification problem may otherwise not provide any clues as to *what* precisely the network has learnt.

During application of the backprop algorithm, it often appears that weights have converged (to finite values). For instance, the difference between consecutive iterates

becomes less than any given tolerance. What is perhaps not so well-known is that weights may very well be diverging, but at a *logarithmic* (or qualitatively similar) rate, which implies in particular that the differences between iterates will indeed go to zero, even if the actual values aren't.

To see why a logarithmic growth may be possible, consider the easiest example of the above setup, namely the gradient minimization of

$$(\theta(x) - 1)^2,$$

where $\theta(x) = \tanh(x)$. (In more realistic examples, if one of the coordinates x_i dominates, one might expect a behavior for x_i that is close to the one for this equation.) Here the associated gradient system is

$$\dot{x} = \alpha(1 - \theta(x))(1 - \theta(x)^2),$$

where $\alpha > 0$ is the "learning rate". Rescaling time, we may assume that $\alpha = 1$. The right hand side is always positive, so solutions go to $+\infty$, as they should since the only local (and hence global) minimum is there. We make the substitution $y := \theta(x)$. Then y satisfies the differential equation

$$\dot{y} = (1 - y^2)^2(1 - y) = (1 + y)^2(1 - y)^3,$$

which for $y \simeq 1$ (i.e. x large) is approximated by

$$\dot{y} = 4(1 - y)^3.$$

The solution of this latter equation is up to a constant equal to

$$1 - \frac{1}{\sqrt{8t + 1}}.$$

It follows that

$$x(t) = \tanh^{-1}(y(t)) = \log\left(\frac{1 + y(t)}{1 - y(t)}\right)$$

has asymptotically the form $a + b \log t$, as claimed.

Of course the above argument is not a rigorous proof, and in any case the general situation (many inputs and outputs, hidden neurons) is much more complicated. But we tried nonetheless some experiments in which we measured the rate of change of weights $w(t)$ during backprop. If log growth occurs, the expression

$$t[w(t + 1) - w(t)]$$

which approximates $t.\dot{w}(t)$, would have to become eventually constant. Figure 5 shows typical results (in this case, on the "XOR" problem,) that seem to confirm the above heuristic argument. (This graph was obtained by Hasanat Dewan, from the Rutgers Computer Science Department, who tested the hypothesis numerically.) We plan much more experimentation along these lines in the future.

If the "log rule" turns out to have wide applicability, it may be used eventually to speed learning. Once that diverging weights are in logarithmic growth mode, it is possible to extrapolate their limiting values, or more importantly for implementations, one may increase these values while preserving whatever proportions are needed, in order to improve network performance.

We close with an observation which shows that, under many circumstances, some weights will indeed diverge. (A more general result, for multilayer nets and corresponding recognizable sets, should also be possible.)

Lemma. With the notations of the previous section, assume that the rows of the matrix A are all contained in a half space and are linearly independent (classical "linear separability" condition). Then the gradient system

$$\dot{x} = -\alpha(\nabla E)'(x)$$

where

$$E(x) = \|e - TAx\|^2,$$

has no bounded solutions.

Proof. Let $\nu \neq 0$ be so that each coordinate of $A\nu$ is nonnegative and at least is strictly positive. Then

$$\nabla E(x).\nu = -2\sum_i (1 - \theta(\sum_j a_{ij}x_j))\theta'(\sum_j a_{ij}x_j))A_i\nu$$

(A_i denotes the i-th row of A) is negative for all x, since at least one of the summands is strictly positive. Thus ∇E is always nonzero, so E decreases strictly along every trajectory. The rest is a standard Lyapunov-type argument. Indeed, assume that $x(\cdot)$ is a trajectory that stays in a compact, and let $\rho > 0$ be the infimum of the values $E(x(t))$ along this trajectory. There is a sequence of times t_i so that

$$E(x(t_i)) \to \rho$$

and we may assume without loss that the sequence $\{x_i\}, x_i := x(t_i)$, converges to some (finite) ξ. Thus by continuity $E(\xi) = \rho$. But a trajectory starting at ξ will have smaller E, so

$$E(x(t_i + \varepsilon)) < \rho$$

for any $\varepsilon > 0$ and i large enough, contradicting the choice of ρ. ∎

4 Conclusions

We have shown that even in the simplest possible case it is possible to have a nontrivial local minima structure, and we have made some remarks about the rate of growth of diverging coefficients. As suggestions for further work, we wish to state the following:

- Study the local minima structure for more realistic examples. How many local minima may appear? For instance, can there be a number exponential on the number of neurons?
- Are there special characteristics of backprop problems corresponding to "practical", –for instance geometric,– tasks that make the local minima structure simpler in those cases?
- Is it really worth using relaxation methods?
- Is the log rule verified in large scale examples? If so, can it be usefully employed to extrapolate from limited learning?

5 References

[Ar87] Arbib, M.A., *Brains, Machines, and Mathematics, Second Edition*, Springer, 1987.

[Hi87] Hinton, G.E., "Connectionist learning procedures," Technical Report CMU-CS-87-115, Comp.Sci. Dept., Carnegie-Mellon University, June 1987.

[KGV83] Kirkpatrick,S., C.D.Gelatt, and M.P.Vecchi, "Optimization by simulated annealing," *Science* **220**(1983): 671-680.

[KH88] Kung, S.Y. and J.N.Hwang, "An algebraic projection analysis for optimal hidden units size and learning rates in backpropagation learning," in *Proceedings of the IEEE Int'l Conf.on Neural Networks*, San Diego, 1988.

[PDP] Rumelhart, D.E., and J.L. McClelland, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, MIT Press, 1986, volume 1.

[SS85] Sontag, E.D., and H.J. Sussmann, "Image restoration and the annealing algorithm", *Proc. IEEE Conf. Dec. and Control*, 1985, pp.768-773.

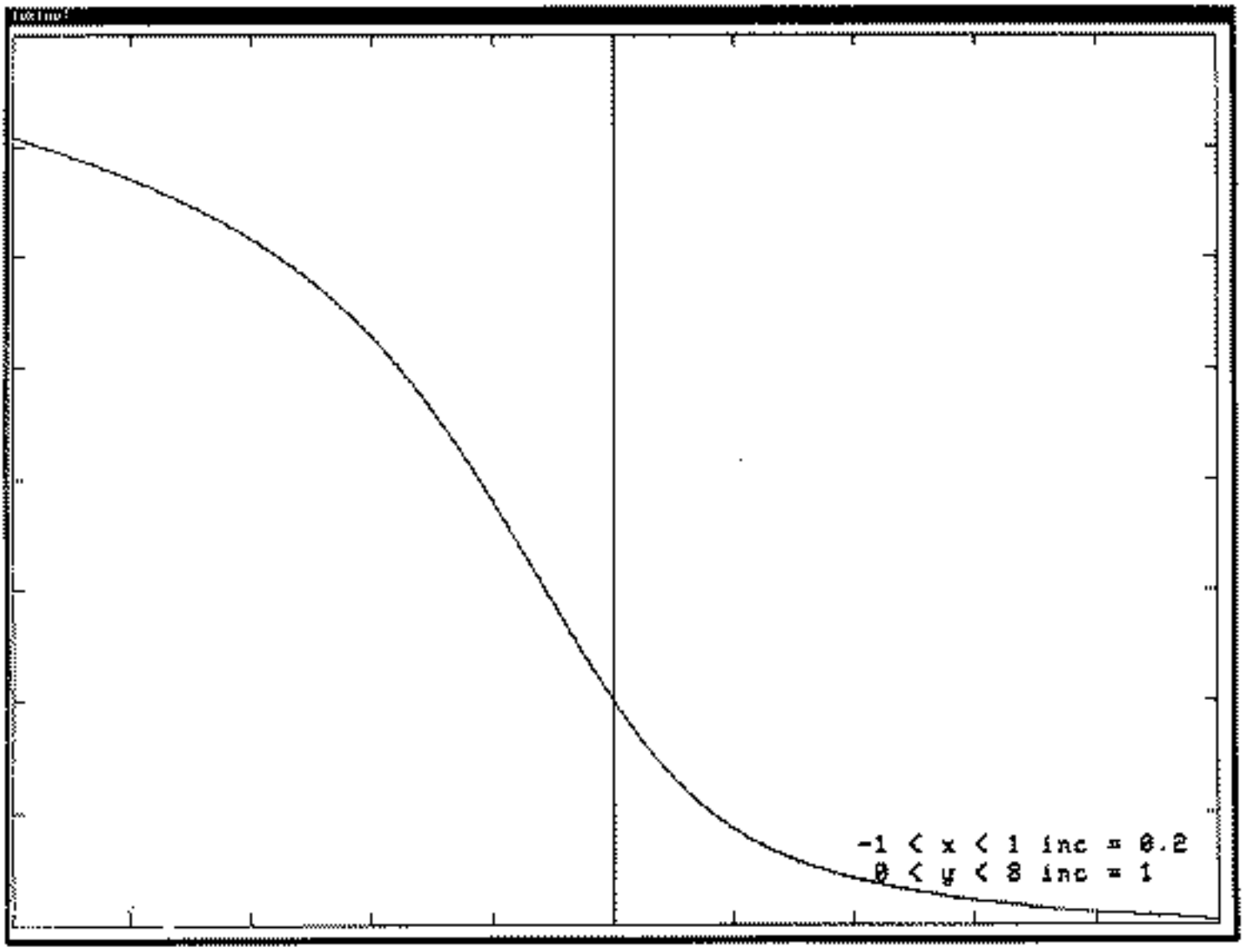


Figure 1

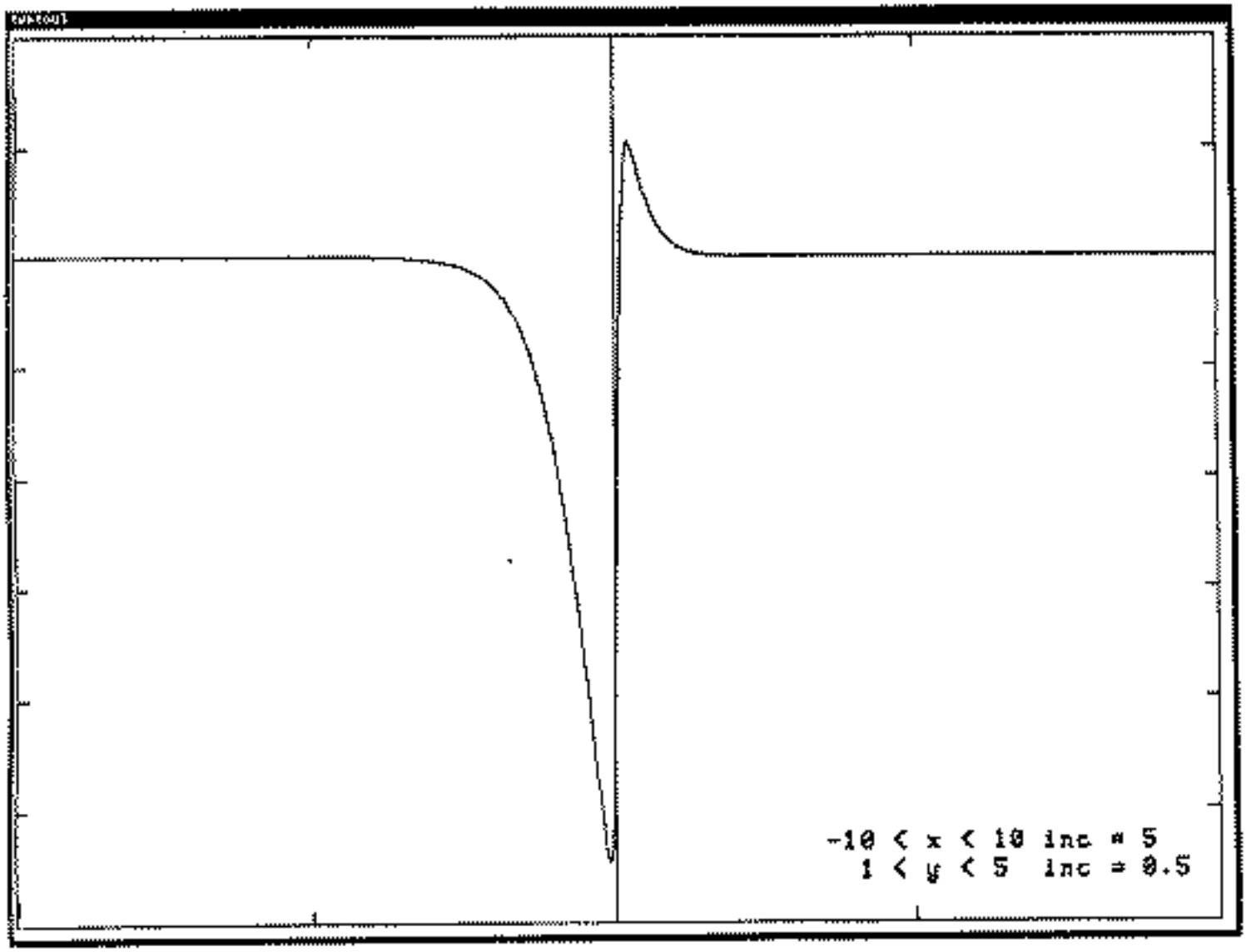


Figure 2

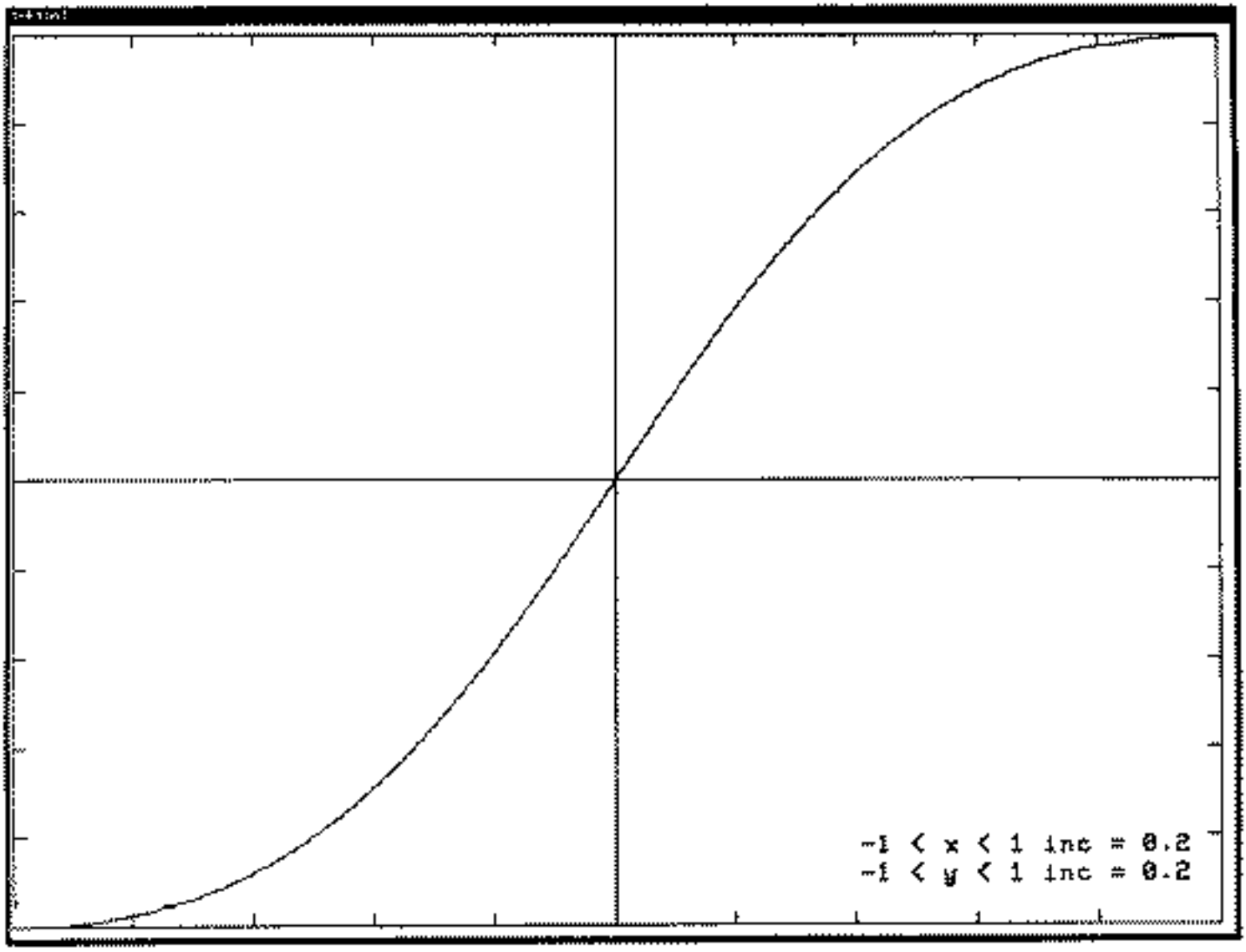


Figure 3

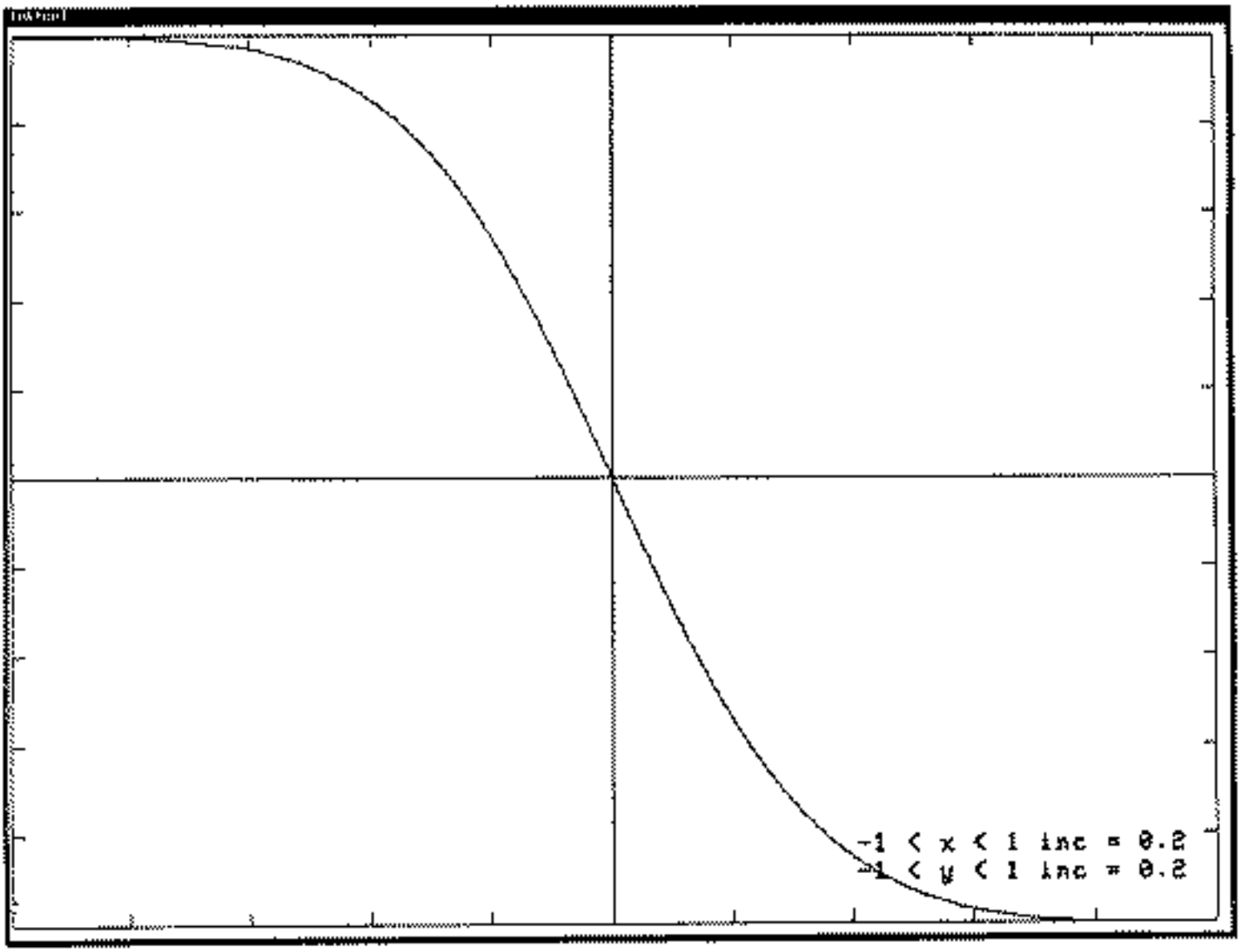
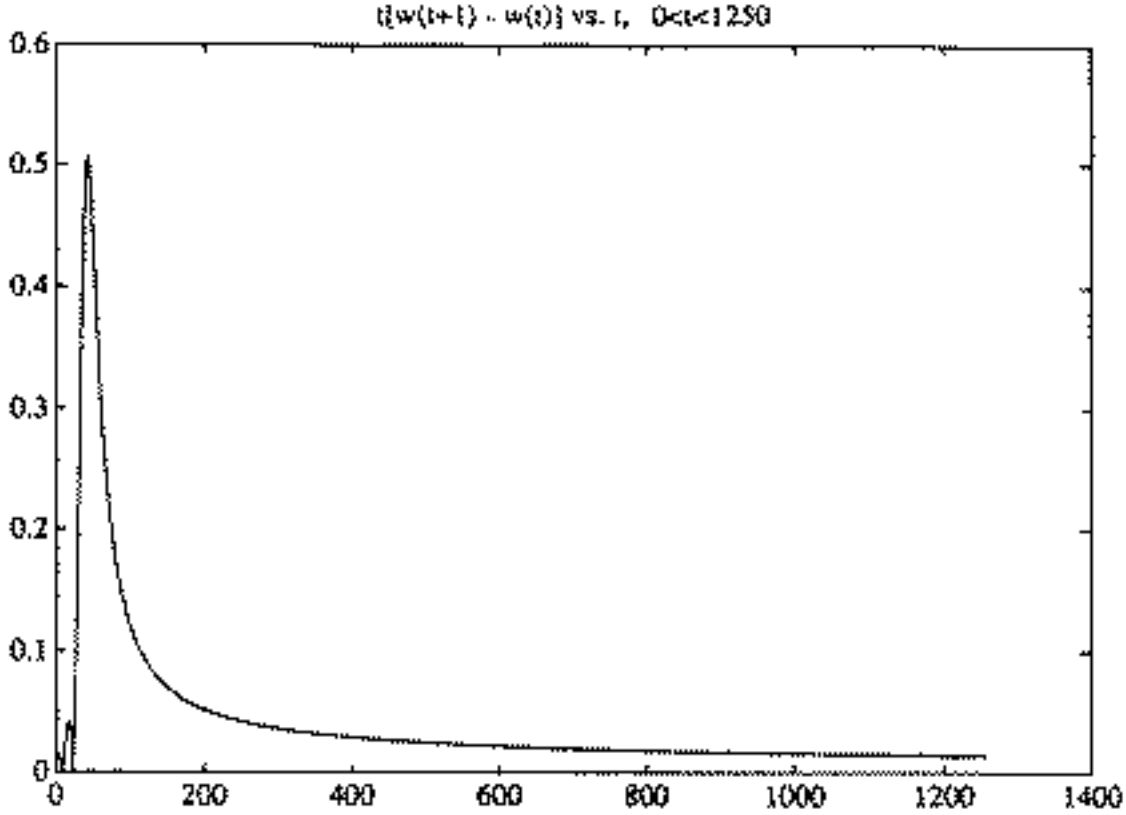


Figure 4


t[w(t+1) - w(t)] vs. t, 0<t<1250
0.6
0.5
0.4
0.3
0.2
0.1
0
0
200
400
600
800
1000
1200
1400

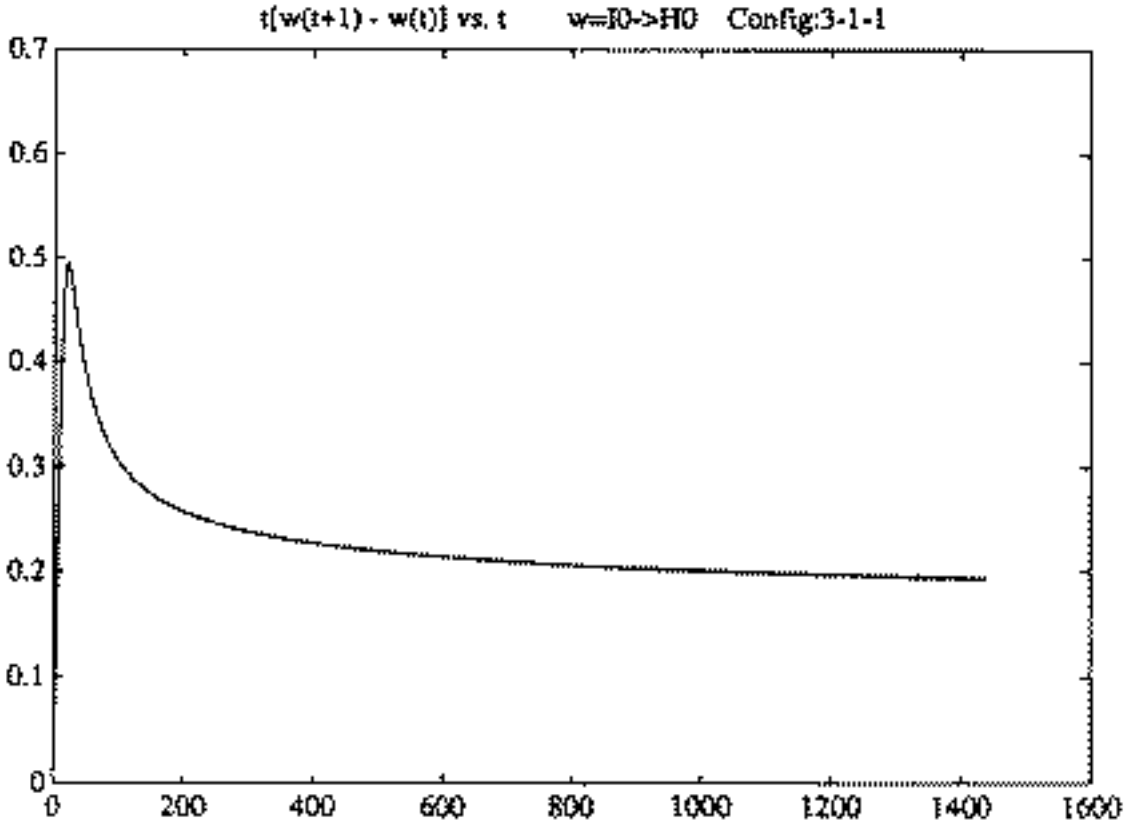

t[w(t+1) - w(t)] vs. t w=I0->H0 Config:3-1-1
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
200
400
600
800
1000
1200
1400
1600

Figure 5