Communicated by Andrew Barto

The Bootstrap Widrow–Hoff Rule as a Cluster-Formation Algorithm

Geoffrey E. Hinton
Steven J. Nowlan
Department of Computer Science, University of Toronto,
10 King's College Road, Toronto M5S 1A4, Canada

An algorithm that is widely used for adaptive equalization in current modems is the "bootstrap" or "decision-directed" version of the Widrow–Hoff rule. We show that this algorithm can be viewed as an unsupervised clustering algorithm in which the data points are transformed so that they form two clusters that are as tight as possible. The standard algorithm performs gradient ascent in a crude model of the log likelihood of generating the transformed data points from two gaussian distributions with fixed centers. Better convergence is achieved by using the exact gradient of the log likelihood.

1 Introduction

Modems are used to transmit bits along analog lines. Since the bandwidth is limited, it is impossible to transmit square pulses so the contribution that each individual bit makes to the transmitted signal is necessarily extended in time as shown in Figure 1. The received signal is strobed at the points where the individual bits make their maximum contributions, but the "intersymbol interference" from nearby bits can cause the strobe values to have the wrong sign. So a simple threshold decision rule based on the strobe value alone is imperfect. Better decisions can be made by taking into account the local temporal context.

A simple way to use local context is to have delay taps that represent the value of the received signal at nearby strobe times.[1] The delay taps can be used as the input lines to a linear unit that forms a weighted combination of a strobe value with nearby values before the thresholding operation is applied. If the desired outputs of the thresholding operation are known, the weights can be trained by using a supervised learning procedure such as Widrow–Hoff (Widrow and Hoff 1960). When the output should be above (below) threshold, we assume that the desired output value is +1 (−1). Unfortunately, this requires that a known bit-string be transmitted.

[1]Fractional delay times that sample more frequently than the strobe period are also used.

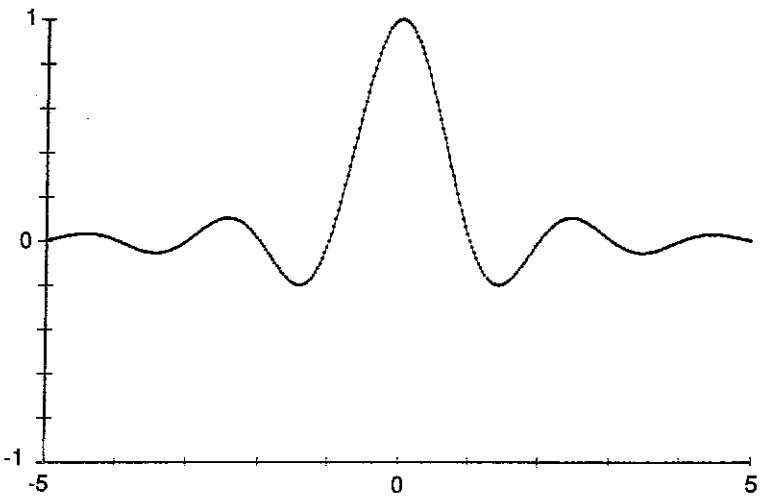

Figure 1: Contribution of an individual bit to the transmitted signal.

If we wish to avoid sending a known signal, or if we wish to continually adapt to the properties of the line, then we need some other way of deciding what the correct output of the decision process should have been. In the "bootstrapping" or "decision-directed" algorithm (Lucky 1966), the actual output of the linear unit is thresholded at zero and it is assumed that the decision is correct. So whenever the actual output is above zero, we assume that the desired output is +1 and we adjust the weights accordingly. Figure 2 uses a very simple example to show why this procedure works. If the initial weights cause only a few cases to be misclassified, the learning that takes place for the correctly classified cases will eventually correct the errors, even though the learning on the wrongly classified cases moves the weights in the wrong direction.

The bootstrapping algorithm works well in practice (Widrow and Stearns 1985; Qureshi 1985) and it is one of the most important current applications of neural networks. However, there seems to be little theoretical justification for using the thresholded actual output to determine the desired output. The aim of this paper is to provide a justification by showing that the bootstrapping algorithm can be viewed as an unsupervised cluster-formation algorithm that has a close relationship to competitive learning. Like competitive learning, the bootstrapping algorithm makes a "winner-take-all" decision in deciding which cluster

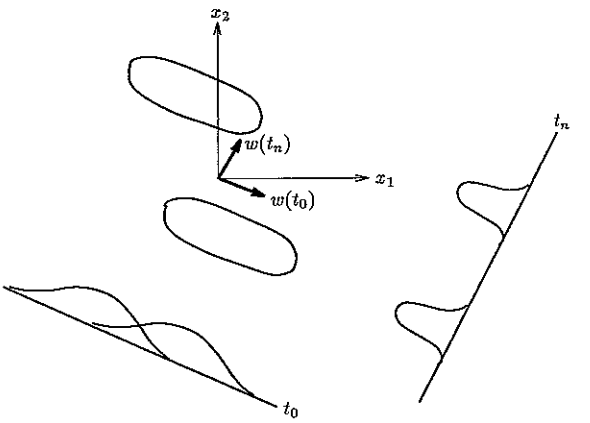

Figure 2: The x_1 and x_2 axes represent the values on two delay taps that are used as input to a linear unit. The two ellipses represent the distributions of two clusters of input vectors. The thick line labeled $w(t_0)$ represents the initial weight vector. When the two clusters are projected onto this weight vector, they yield the two overlapping gaussians shown beneath. With learning, the weight vector rotates so that the projection of the clusters forms two well-separated gaussians.

should be given responsibility for a given data point. Its convergence can be improved by using a statistically more reasonable "soft" decision in which the responsibility of a cluster for a data point depends on the relative probability of generating the data point from the cluster.

2 The Objective Function for the Bootstrapping Procedure

One simple form of competitive learning works by minimizing the squared distance between each data point and the nearest cluster center by moving the cluster center toward the data point by an amount proportional to their separation (Hinton 1989). If we view the *output* of the linear unit as a data point, the bootstrap algorithm minimizes the same objective function by moving the data point toward the center of the nearest cluster.

Figure 2 shows that the bootstrapping procedure adjusts the weight vector so that the actual output of the linear unit forms two sharp clusters, one centered around +1 and the other centered around −1. This suggests the following interpretation of what the bootstrapping is really achieving.

We start with the prior belief that the *output* of the linear unit can be modeled by two gaussian distributions, one centered at -1 (for the 0 bits) and the other centered at $+1$ (for the 1 bits). We want to make the outputs of the linear unit fit this mixture of gaussians model as well as possible. This can be accomplished by modifying the weights of the linear unit and the parameters of the gaussians to maximize the log likelihood of generating the observed output values from the mixture of gaussians model:

$$\log L = \sum_t \log \left(\frac{\rho_1}{\sqrt{2\pi}\sigma_1} e^{-(x^t-\mu_1)^2/2\sigma_1^2} + \frac{\rho_2}{\sqrt{2\pi}\sigma_2} e^{-(x^t-\mu_2)^2/2\sigma_2^2} \right) \tag{2.1}$$

where x^t is the output value of the linear unit for the tth training case, ρ_1 and ρ_2 are the proportions of the two gaussians in the mixture, σ_1 and σ_2 are their variances, and μ_1 and μ_2 are their means.

Later we will consider how to adapt the parameters of the two gaussians, but initially we will assume that these parameters are fixed and only the weights of the linear unit can be adapted. To perform the online version of gradient ascent in $\log L$, we need to change each weight, w_i, by

$$\Delta w_i^t = \epsilon \, a_i^t \frac{\partial \log L}{\partial x^t} \tag{2.2}$$

where ϵ is the learning rate, a_i^t is the activity level of the ith delay tap, and x^t is the output of the linear unit in case t.

A crude way to deal with the sum of two exponential terms in equation 2.1 is to ignore the smaller one. If we assume that the two gaussians have equal variances and equal mixing proportions, this amounts to ignoring the possibility that an output value x^t could have been generated from the gaussian that is farther away. With this crude simplification and the assumption that $\mu_1 < \mu_2$ we get a very simple expression for the derivative of the log likelihood

$$\frac{\partial \log(L)}{\partial x^t} \approx \begin{cases} \frac{1}{\sigma^2}(\mu_1 - x^t) & x^t < 1/2(\mu_1 + \mu_2) \\ \frac{1}{\sigma^2}(\mu_2 - x^t) & \text{otherwise} \end{cases} \tag{2.3}$$

The $1/\sigma^2$ term simply modifies the learning rate, so if we set $\mu_1 = -1$ and $\mu_2 = +1$ and substitute equation 2.3 into equation 2.2 we get exactly the bootstrap Widrow–Hoff procedure. The simplification used is identical to the simplification used in "hard" competitive learning in which the weights that represent the "center" of a competitive unit are regressed toward the current input vector if and only if that competitive unit wins the competition. This is equivalent to treating the competitive units as gaussians of fixed variance and only adapting the center of the gaussian most likely to have generated the data.

3 A Correct Maximum Likelihood Learning Procedure

We continue to assume that the gaussians have equal variances and equal mixing proportions, but we now take into account the fact that any given output value could be generated by either gaussian. Equation 2.1 then yields

$$\frac{\partial \log L}{\partial x^t} = \frac{\lambda}{\sigma^2}(\mu_1 - x^t) + \frac{1-\lambda}{\sigma^2}(\mu_2 - x^t) \tag{3.1}$$

where

$$\lambda = \frac{1}{1 + e^{-[2x^t(\mu_1-\mu_2)+\mu_2^2-\mu_1^2]/2\sigma^2}} \tag{3.2}$$

Comparing equation 3.1 with equation 2.3, we see that the hard decision between the two gaussians has been replaced by a soft, sigmoid function which varies smoothly from 0 to 1 with a value of 0.5 at the midpoint of the two gaussians. If $\mu_1 = -\mu_2 = -1$ and $\sigma = 1$, the exponent in the expression for λ is simply $2x^t$.

The weakness of the hard decision used by the bootstrapping procedure is apparent when we realize that when the output values are very near 0 they are most likely to have the wrong sign because these are the cases when the hard decision is most likely to be incorrect (assuming that the means of the two gaussians are symmetric about 0). Yet it is in these cases that the weights are changed the most. In the "soft" algorithm, with its sigmoid decision function, the two terms approximately cancel out for output values near 0, so it makes only small weight changes in these highly ambiguous cases.

Following this line of reasoning, we might expect that if there is enough noise and distortion to force the outputs to be frequently in the region near 0, the "soft" algorithm will outperform the "hard" algorithm. Simulation results support this conclusion. Figure 3 shows a set of typical simulation results. The curves in this figure show the mean squared error (in dB) versus the number of updates for the "soft" model with several different values for σ, the variance of the two gaussians. Larger values of σ correspond to a shallower slope for the sigmoidal decision function. $\sigma = 0$ corresponds to the "hard" decision rule. The mean squared error is decreased most rapidly initially by using relatively large values of σ, showing the superiority of the soft algorithm when the output decisions are prone to error. The hard algorithm eventually catches up to the soft algorithm as the equalizer adapts and the output values become close to $+1$ (μ_2) or -1 (μ_1) most of the time. The most rapidly converging algorithm, labeled by **var** in the figure, continuously reestimated the variance while using the soft decision rule. This amounts to adaptively adjusting the learning rate (see below).

In modem equalization it is crucial to be able to adapt rapidly to sudden fluctuations in the transmission line. The use of a "soft" decision

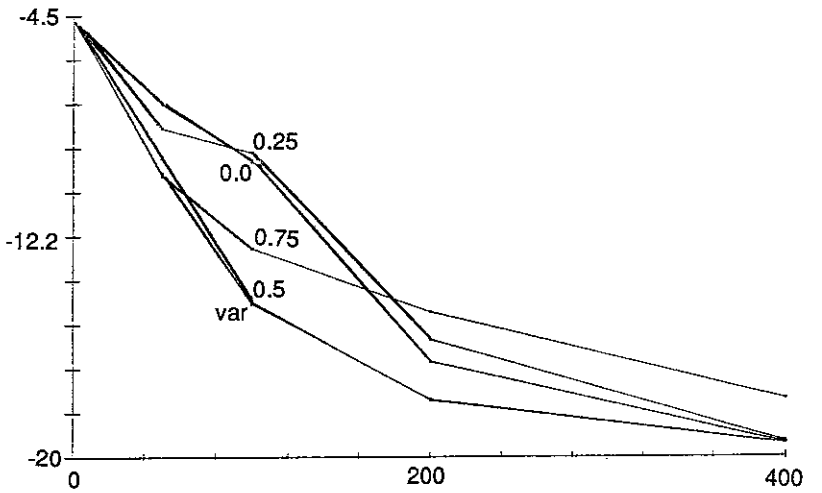

Figure 3: Mean squared error versus the number of updates for a signal with moderate distortion.

rule in the bootstrap learning procedure allows more rapid adaptation to sudden fluctuations by discounting the error terms that are most likely to be incorrect.

4 Adapting the Parameters of the Gaussians

If the means of the gaussians were allowed to adapt, both means and all the output values would converge on the same value. However, the mixing proportions and the variances can be adapted to maximize log likelihood. The most tractable step in this direction is to assume the mixing proportions are fixed and equal and to allow the variances to adapt subject to the constraint that they remain equal. One method of adapting the variances would be to change σ by an amount proportional to the derivative of the log likelihood. A faster method, that we used in the simulation labeled **var** in Figure 3, is an incremental version of the EM algorithm (Dempster et al. 1976). The batch version of EM simply sets σ to a value guaranteed to yield higher likelihood (given the current output values) after a batch of training cases. The incremental version uses an exponential decay factor to weight previous cases, and then applies EM after every training case. The actual update rule then becomes

$$\sigma^2(t+1) = \kappa\sigma^2(t) + (1-\kappa)\left[\lambda(x^t-\mu_1)^2 + (1-\lambda)(x^t-\mu_2)^2\right] \qquad (4.1)$$

where κ is a decay rate slightly less than 1 for discounting past data, and λ is defined in equation 3.2. This incremental method does not

require a learning rate for the variance adaptation. The decay rate for the exponential averaging of past data is based on the degree of stationarity of the data, not on properties of the learning algorithm.

5 The Extension to Multilayer Networks

Our analysis of the bootstrapping procedure used for adaptive equalization can be extended to more complex networks and gives rise to a new class of unsupervised learning procedures in which the objective of forming tight clusters is used to generate an error signal that can be backpropagated through layers of nonlinear units. To prevent all the input vectors from being mapped to the same output vector, we can fix the cluster centers in advance, or we can use a scale-invariant objective function that takes the distance between cluster centers into account when evaluating the tightness and thus eliminates the ability to improve the objective function by simply moving the cluster centers closer together. To prevent arbitrary mappings of input vectors into clusters, we can insist on using relatively simple functions. For example, we can assume that the log likelihood of a function is proportional to the sum of the squares of the weights that it uses and we can then trade-off cluster tightness against the prior log likelihood of the function. A more sophisticated version of this idea would be to estimate the complexity of the function by fitting a mixture of gaussians model to the set of weight values and using the combined description length of the mixture model and the weights given that model as an upper bound on the complexity. In this case, we would be trading off the tightness of the clusters of output values against the tightness of the clusters of weight values.

Bridle has independently developed a similar approach to unsupervised cluster-formation in multilayer networks (John Bridle, unpublished research note SP4-RN66, October 1988).

6 Discussion

Learning procedures for neural networks are often designed by an intuitive leap (Durbin and Willshaw 1987; Crick and Mitchison 1983). This is particularly true of the unsupervised learning procedures. Those that actually work in practice, are usually found to be an approximation to steepest ascent in some sensible statistical measure such as log likelihood or mutual information (Durbin et al. 1989; Hinton and Sejnowski 1986). The bootstrapping algorithm is no exception. This adds further support to the idea that future unsupervised algorithms should be designed by explictly differentiating a sensible objective function, and then making approximations to achieve easy implementation.

Acknowledgments

We thank Yann Le Cun and John Bridle for helpful discussions. This research was funded by a grant from the Ontario Information Technology Research Center.

References

Crick, F., and Mitchison, G. 1983. The function of dream sleep. *Nature (London)* **304**, 111–114.

Dempster, A. P., Laird, N. M., and Rubin, D. B. 1976. Maximum likelihood from incomplete data via the EM algorithm. *Proc. R. Stat. Soc.* 1–38.

Durbin, R., Szeliski, R., and Yuille, A. 1989. An analysis of the elastic net approach to the travelling salesman problem. *Neural Comp.* **1**, 348–358.

Durbin, R., and Willshaw, D. 1987. The elastic net method: An analogue approach to the travelling salesman problem. *Nature (London)* **326**, 689–691.

Hinton, G. E. 1989. Connectionist learning procedures. *Artificial Intelligence* **40**, 185-234.

Hinton, G. E., and Sejnowski, T. J. 1986. Learning and relearning in Boltzmann machines. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*, D. E. Rumelhart, J. L. McClelland, and the PDP group, eds. MIT Press, Cambridge, MA.

Lucky, R. W. 1966. Techniques for adaptive equalization of digital communications systems. *Bell Syst. Tech. J.* **45**, 255–286.

Qureshi, S. U. H. 1985. Adaptive equalization. *Proc. IEEE* **73**, 1349–1387.

Widrow, B., and Hoff, M. E., Jr. 1960. Adaptive switching circuits. *IRE WESCON Convention Rec.* 96–104.

Widrow, B., and Stearns, S. D. 1985. *Adaptive Signal Processing*. Prentice-Hall, Englewood Cliffs, NJ.

Received 15 March 90; accepted 3 May 90.