Performance Surfaces of a Single-Layer Perceptron

JOHN J. SHYNK, MEMBER, IEEE

Abstract—**A perceptron learning algorithm may be viewed as a steepest-descent method whereby an instantaneous performance function is iteratively minimized. This paper describes an appropriate performance function for a widely-used perceptron algorithm and shows that the update term of the algorithm is the gradient of this function. An example is given of the corresponding performance surface based on Gaussian assumptions and it is shown that there is an infinity of stationary points. The performance surfaces of two related performance functions are also examined, from which alternative perceptron algorithms can be derived.**

I. Introduction

THE perceptron is a linear combiner that quantizes its output to one of two discrete values. Its connection weights and threshold (or bias) can be fixed or adapted using a variety of different learning algorithms. The original perceptron convergence procedure for adjusting the weights was developed by Rosenblatt [1], who proved that if the inputs presented from two classes are linearly separable (i.e., they fall on opposite sides of a hyperplane), then the perceptron algorithm converges and it positions the decision boundary between these two classes. The perceptron with adaptive weights is also often referred to as Adaline (for *ada*ptive *l*inear *ne*uron) [2], [3]. The LMS (least-mean-square) algorithm developed by Widrow and Hoff [2], [4] is analogous to Rosenblatt's algorithm, although it has been used primarily for a linear combiner without quantization, minimizing the difference between a desired response and the linear output.

Multilayer perceptrons (or Madelines [3]) are feedforward neural networks with one or more layers of Adalines (perceptron nodes) between the input and output nodes. Two- and three-layer perceptrons can represent quite complicated decision regions whereas a single-layer perceptron can only separate regions by a hyperplane. With the development of new training algorithms (e.g., backpropagation [5] and Madeline rule II [3]), it has become practical to use these multilayer perceptrons to solve decision problems with sophisticated boundaries. This paper deals only with a single-layer perceptron, but many of the ideas carry over to multilayer perceptrons.

For a single-layer perceptron [6], the input signals $\{x_k\}$ are scaled by a set of adjustable weights $\{w_k\}$ to generate an intermediate output signal y, which is then processed by a hard limiter, resulting in the quantized binary output $y_q(\pm 1)$. This binary output is then compared to the desired response d_q, which is also a binary signal, generating an error e_q that is used in a feedback strategy to adapt the weights. An adaptive algorithm based on the method of steepest descent [4] is generally used to modify the weights so as to minimize a specific performance criterion. The input signals can be binary-valued or they can be drawn according to a continuous distribution. In some cases, the hard limiter is replaced by a smooth nonlinearity, such as a sigmoidal function [6].

A single-layer perceptron essentially operates as a pattern classification device whereby the N-dimensional vector space, represented by the filter input signals, is partitioned by a hyperplane into two subspaces [3]. For the case with two input signals ($N = 2$), the two-dimensional space is defined by the x_1-x_2 axes and the separating hyperplane is a straight line which partitions the space into two half-planes. This line can be expressed as $x_2 = -(w_1/w_2)\,x_1 - (b/w_2)$, whose slope is determined by the *relative* values of weights w_1 and w_2. The ordinate crossing is specified by b, which is a bias term that may also be adaptive. As an example, when $b = 0$ and $w_1 = -w_2$, the partition is a line passing through the origin with a slope of 45°. The perceptron is trained to identify this decision boundary, so that for any two input samples x_1 and x_2, it will correctly decide to which class the sample pair belongs. The location of the line is determined by the *correlation* properties of the input signals $\{x_k\}$ and the desired response d_q.

This paper reviews certain aspects of the perceptron and describes a performance function for a widely-used perceptron algorithm [6], [7], which is viewed as a steepest-descent method. It is shown that the performance function is not the usual mean-square-quantized error, although it does have the same minimum points. In Section II, the standard perceptron algorithm is presented with a possible model for the desired response, that can be useful for analytical purposes. In Section III, the associated performance function which the algorithm attempts to minimize is presented. The convergence properties of this algorithm have been examined previously in [8] and [9]. Two related performance functions are also discussed and the resulting gradients lead to well-known alternative perceptron algorithms. Based on certain Gaussian assumptions, analytical expressions are derived for all three of these performance functions and their minimum points. In Section IV, examples of the performance surfaces associated

Manuscript received March 9, 1990; revised May 14, 1990. This work was supported by the University of California MICRO Program, Rockwell International, and Applied Signal Technology, Inc.

The author is with the Center for Information Processing Research, Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106.

IEEE Log Number 9037205.

1045-9227/90/0900-0268$01.00 © 1990 IEEE

with these performance functions are shown for comparison. Computer simulations are also given that demonstrate the convergence properties of the adaptive algorithms. Conclusions are outlined in Section V.

II. Perceptron Learning Algorithm

The standard perceptron algorithm developed by Rosenblatt has the following recursive form [1], [6] where $W(n)$ and $X(n)$ are N-dimensional column vectors with components $\{w_k(n)\}$ and $\{x_k(n)\}$, respectively, and n is the discrete-time index:

$$W(n+1) = W(n) + 2\mu e_q(n)X(n). \tag{1}$$

The positive step-size μ controls the convergence rate and steady-state properties of the algorithm. The error term $e_q(n)$ is the difference between the desired value $d_q(n)$ and the quantized output $y_q(n)$, where $\operatorname{sgn}(\cdot)$ is the sign function

$$e_q(n) = d_q(n) - y_q(n) = d_q(n) - \operatorname{sgn}\big(y(n)\big). \tag{2}$$

The unquantized output $y(n)$ is the inner product between the weights and the input signals

$$y(n) = W^T(n)X(n) \tag{3}$$

where the superscript T denotes transpose.

Although $y(n)$ is quantized as part of the perceptron structure, $d_q(n)$ is usually provided directly as a binary signal. However, one can always view $d_q(n)$ as being a quantized version of some *underlying* process $d(n)$, according to $d_q(n) = \operatorname{sgn}(d(n))$ [8], [9]. This representation is completely general and it is a useful interpretation that lends itself to a convenient analytical form. Clearly, $d(n)$ is correlated with $X(n)$ (otherwise, the perceptron signals have not been chosen properly) and it can often be represented as a function (possibly nonlinear) of the elements of $X(n)$. One useful case that will be considered later is such that $d(n)$ is a linear combination of the elements of $X(n)$:

$$d(n) = F^T(n)X(n) \tag{4}$$

where $F(n)$ is defined in a manner similar to $W(n)$, but with components $\{f_k(n)\}$. For the example of the line passing through the origin at an angle of 45°, $d(n) = x_2(n) - x_1(n)$ and $F = [-1 \quad 1]^T$. With this definition, the region above the line corresponds to $d_q(n) = +1$, and the region below the line corresponds to $d_q(n) = -1$.

The algorithm in (1) has a form that is similar to that of the LMS (least-mean-square) algorithm [4], except that the filter output is quantized to generate $y_q(n)$, and the desired response is binary. The LMS algorithm is based on the method of steepest descent and it attempts to minimize an instantaneous estimate of the mean-square error, which is a widely-used performance function. This estimate is simply $e^2(n)$ where $e(n) = d(n) - y(n)$. A steepest-descent algorithm has the following form [4]:

$$W(n+1) = W(n) - \mu\nabla(n) \tag{5}$$

where $\nabla(n)$ is an instantaneous estimate of the gradient ∇ of the performance function. By differentiating $e^2(n)$ with respect to $W(n)$, the gradient estimate for the LMS algorithm is $\nabla(n) = -2e(n)X(n)$, and substituting this result into (5) gives a recursion similar to that in (1). The main difference is that the LMS algorithm is a linear function of the signals, whereas the perceptron algorithm involves a nonlinear function via the hard limiter.

By comparing (1) to the LMS algorithm, it is seen that the perceptron algorithm is *not* derived by minimizing an instantaneous estimate of the mean-square-quantized error, i.e., differentiating $e_q^2(n)$ with respect to $W(n)$ does not lead to the update term $\nabla(n) = -2e_q(n)X(n)$. Even if the sign function is replaced with a smooth nonlinearity, the form in (1) is still not obtained. Nevertheless, the perceptron algorithm can still be viewed as a steepest-descent method by defining an appropriate performance function, which is necessarily different from both the mean-square error and the mean-square-quantized error.

III. Perceptron Performance Functions

In this section, three performance functions are presented that can be used to derive perceptron learning algorithms. These functions will be denoted by ξ_m, their gradients by ∇_m, and the minimum points by W_m, all of which are independent of n. In general, useful analytical expressions are not easily derived for these quantities unless some assumptions are made concerning the probability distributions of $\{x_k(n)\}$ and $d_q(n)$. The case where $\{x_k(n)\}$ is jointly Gaussian and $d_q(n)$ is arbitrary is presented first and then the special case where the underlying process $d(n)$ is also Gaussian, generated according to (4), is considered.

A. Performance Function Based on $|y(n)|$

Consider the instantaneous performance function

$$\xi_1(n) = 2\big|y(n)\big| - 2d_q(n)y(n) \tag{6}$$

where $y(n)$ and $d_q(n)$ were previously defined. Differentiating (6) with respect to $W(n)$, the instantaneous gradient is

$$\begin{aligned}\nabla_1(n) &= 2\operatorname{sgn}\big(y(n)\big)X(n) - 2d_q(n)X(n)\\ &= -2e_q(n)X(n)\end{aligned} \tag{7}$$

and substituting this result into (5), the algorithm in (1) is obtained. Therefore, if the perceptron algorithm is viewed as a steepest-descent method, (6) is the instantaneous performance function which the algorithm attempts to minimize. Although it is not immediately obvious from this expression, it can be shown that $\xi_1(n)$ is a nonnegative function—a property expected of a valid performance function.

By observing that $|y(n)| = \operatorname{sgn}\big(y(n)\big)y(n)$, (6) can be rewritten as

$$\xi_1(n) = 2y_q(n)y(n) - 2d_q(n)y(n) = -2e_q(n)y(n). \tag{8}$$

Recall that $e_q(n)$ is *not* a quantized form of $e(n)$ but is, instead, the difference between two quantized signals as defined in (2). Equation (8) represents an instantaneous estimate of the following performance function:

$$\xi_1 = -2E[e_q(n)y(n)] \tag{9}$$

where E denotes a statistical expectation. This function is quite different from the mean-square-quantized error (which is discussed in detail in the next section). Observe that (8) involves signals that are accessible by the algorithm, i.e., it does not explicitly require knowledge of the underlying process $d(n)$.

In order to evaluate (9), it is necessary to make some assumptions about the distributions of $\{x_k(n)\}$. For convenience, assume that $X(n)$ is a Gaussian random vector with a zero mean and a correlation matrix $R = E[X(n)X^T(n)]$. As a result, $y(n)$ is also a zero-mean Gaussian process, when the weights are fixed at W, and it has a variance of $\sigma_y^2 = W^TRW$. Using the following result for Gaussian random variables [10]:

$$E[y_q(n)y(n)] = E[y^2(n)]/(c\sigma_y) = \sigma_y/c \tag{10}$$

where $c = \sqrt{\pi/2}$, it can be shown that

$$\xi_1 = 2(\sigma_y/c - W^TP_q) \tag{11}$$

where the cross-correlation vector is defined as

$$P_q = E[X(n)d_q(n)]. \tag{12}$$

The corresponding gradient is

$$\nabla_1 = 2[RW/(c\sigma_y) - P_q] \tag{13}$$

and, setting (13) equal to the zero vector, the following expression for the stationary points of ξ_1 is obtained:

$$W_1 = c\sigma_y R^{-1}P_q. \tag{14}$$

This result is a nonlinear function of the weight vector W (because of the form of σ_y). One can readily conclude that the stationary points are not unique because W can be solved for any value of σ_y. As a result, there is an *infinity* of stationary points.

Observe that (11), (13), and (14) are general in that no assumptions have been made about the desired response $d_q(n)$. To continue further, assume that $d_q(n)$ is generated according to $d_q(n) = \text{sgn}(d(n))$, where $d(n)$ is a zero-mean Gaussian process with variance σ_d^2. Then, using a result similar to (10),

$$P_q = P/(c\sigma_d) \tag{15}$$

where

$$P = E[X(n)d(n)]. \tag{16}$$

If $d(n)$ is specified by (4) with $F(n)$ fixed at F, then $\sigma_d^2 = F^TRF$ and $P = RF$. Furthermore, the cross-correlation $\rho_{dy} = E[d(n)y(n)] = F^TRW$ can be defined. Substituting these results into (11), (13), and (14), the following closed-formed expressions are derived:

$$\xi_1 = 2[\sigma_y - (\rho_{dy}/\sigma_d)]/c, \tag{17}$$

$$\nabla_1 = 2R[(W/\sigma_y) - (F/\sigma_d)]/c, \tag{18}$$

and

$$W_1 = (\sigma_y/\sigma_d)R^{-1}P = (\sigma_y/\sigma_d)F. \tag{19}$$

This last expression, in terms of $R^{-1}P$, is very similar to the Wiener solution of the LMS algorithm [4] except for the scalar term (σ_y/σ_d). Because σ_y depends on W, a recursive expression for the variance $\sigma_y^2(n)$ can be derived from the algorithm in (1). By examining this recursion near its convergence, one can determine an asymptotic expression for the output variance which, in turn, can be substituted into (19) [8], [9].

Finally, substituting (19) into (17) [or (14) into (11)], the corresponding minimum value of the performance function is

$$\xi_{1,\min} = 0. \tag{20}$$

Also, by substituting (19) into (18) [or (14) into (13)], it can be shown that the optimal solutions correspond to points where the gradient is zero, as expected.

B. Performance Function Based on $e_q^2(n)$

An instantaneous estimate of the mean-square-quantized error is given by

$$\xi_2(n) = e_q^2(n). \tag{21}$$

The difficulty with this performance function is that its derivative involves an impulse function, i.e., the derivative of the sign function is $2\delta(y(n))$, where δ is the Dirac delta function. As a result, the gradient of (21) is not useful as an update term in a steepest-descent algorithm.

A more practical implementation replaces the hard limiter $\text{sgn}(y(n))$ with a sigmoidal function of the form

$$g(y(n)) = 2f(y(n)) - 1 \tag{22}$$

where [6]

$$f(y(n)) = (1 + e^{-\alpha y(n)})^{-1}. \tag{23}$$

The positive scalar α is used to modify the shape (slope) of $g(y(n))$; as $\alpha \to \infty$, $g(y(n)) \to \text{sgn}(y(n))$. It can easily be shown that the derivative of $g(y(n))$, with respect to $y(n)$, is

$$g'(y(n)) = 2\alpha f(y(n))[1 - f(y(n))]. \tag{24}$$

Figs. 1 and 2 show the plots of $g(y)$ and $g'(y)$, respectively, for three different values of α. Observe that the derivative increasingly weights more heavily on the region about $y = 0$ as α becomes large. Substituting (24) into (21), the instantaneous gradient of the performance function, based on the sigmoidal linearity, is

$$\nabla_2(n) = -4\alpha e_q(n)f(y(n))[1 - f(y(n))]X(n) \tag{25}$$

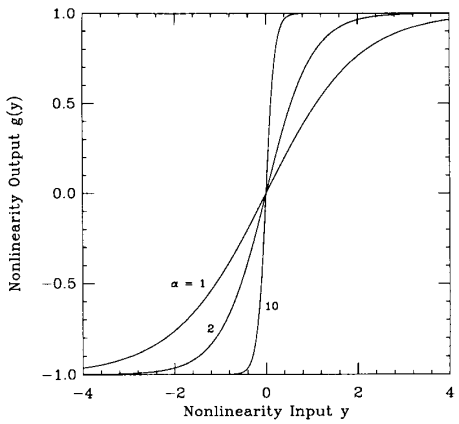

Fig. 1. Sigmoidal nonlinearity for three values of α.

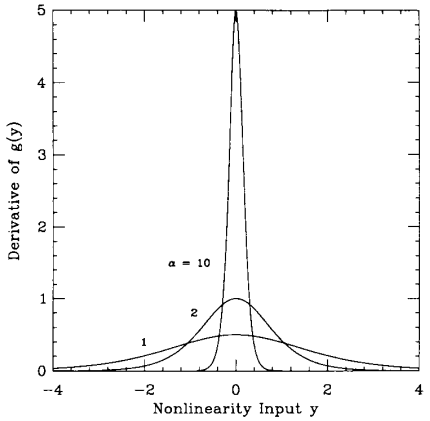

Fig. 2. Derivative of the sigmoidal nonlinearity for three values of α.

and the corresponding perceptron algorithm is

$$W(n+1) = W(n) + 4\mu\alpha e_q(n) f(y(n))[1 - f(y(n))]X(n). \quad (26)$$

Comparing (26) to (1), if one views the standard perceptron algorithm as minimizing (21), then the approximation $2\alpha f(y(n))[1 - f(y(n))] \approx 1$ is being implicitly assumed; this is generally not a reasonable approximation, however, as can be seen in Fig. 2. Note that (26) is actually a form of the back-propagation algorithm [5] applied to a single-layer perceptron.

Returning to the case of the hard limiter (i.e., $\alpha \to \infty$) and taking the expected value of (21),

$$\begin{aligned}\xi_2 &= E[e_q^2(n)] \\ &= E[d_q^2(n) - 2y_q(n)d_q(n) + y_q^2(n)] \\ &= 2 - 2E[y_q(n)d_q(n)]. \end{aligned} \quad (27)$$

Note that $E[d_q^2(n)] = E[y_q^2(n)] = 1$ for any distribution on $y_q(n)$ and $d_q(n)$. If $y(n)$ and $d(n)$ are assumed to be jointly Gaussian as in the previous section, then it is possible to derive the following closed-form expression for ξ_2 using Price's theorem [10]:

$$\xi_2 = 2 - (4/\pi)\sin^{-1}[\rho_{dy}/(\sigma_d\sigma_y)]. \quad (28)$$

Differentiating ξ_2 with respect to W,

$$\nabla_2 = (4/\pi)[R(\rho_{dy}W - \sigma_y^2 F)]/[\sigma_y^2(\sigma_d^2\sigma_y^2 - \rho_{dy}^2)^{1/2}] \quad (29)$$

and setting the numerator of this expression equal to the zero vector and solving for the weights

$$W_2 = (\sigma_y^2/\rho_{dy})F. \quad (30)$$

This result is similar to (19), so there is also an infinity of solutions here and, in fact, these solutions are identical to those defined by (19) because both (σ_y/σ_d) and (σ_y^2/ρ_{dy}) are positive scale factors. The transient convergence properties of the two algorithms can be quite different, however, as demonstrated in Section IV. Finally, the corresponding minimum value of ξ_2 is

$$\xi_{2,\min} = 0 \quad (31)$$

which is the same as (20).

C. Performance Function Based on $y^2(n)$

By examining the previous performance functions, there is another somewhat obvious instantaneous performance function, and this is given by

$$\xi_3(n) = [y(n) - d_q(n)]^2. \quad (32)$$

This expression is similar to that of $\xi_2(n)$, except that $y_q(n)$ has been replaced with $y(n)$. The corresponding instantaneous gradient is

$$\nabla_3(n) = -2[d_q(n) - y(n)]X(n) \quad (33)$$

which is similar to $\nabla_1(n)$, and the algorithm is

$$W(n+1) = W(n) + 2\mu[d_q(n) - y(n)]X(n). \quad (34)$$

Notice that (34) closely resembles the LMS algorithm as described in [4], except that the desired response here is restricted to be a binary signal. This form is also often referred to as the LMS algorithm, and it has been used for perceptron learning [3]. The basic idea behind the performance function in (32) is not only to match the sign of $y(n)$ with that of $d_q(n)$, but also to scale the weights such that $y(n)$ has a variance of $\sigma_y^2 = c^{-2}$ (as shown below). This places an additional restriction on the stationary points and, in fact, there is only one such point. The quantized output $y_q(n)$ is no longer needed in the algorithm but it is still an important part of the perceptron because it provides the decisions once learning is complete.

Taking the expected value of (32),

$$\xi_3 = W^T R W - 2W^T P_q + 1, \tag{35}$$

the corresponding gradient is

$$\nabla_3 = -2(P_q - RW), \tag{36}$$

and the unique stationary point is specified by

$$W_3 = R^{-1} P_q. \tag{37}$$

The minimum value of ξ_3, obtained by substituting (37) into (35), is

$$\xi_{3,\min} = 1 - P_q^T R^{-1} P_q = 1 - P_q^T W_3. \tag{38}$$

Observe that the results in (35)–(38) are general for any distribution on the input and desired response. In fact, they are completely analogous to that derived in [4] for the LMS algorithm for an arbitrary desired response. If the same Gaussian assumptions made before are used here, then

$$\xi_3 = \sigma_y^2 - 2\rho_{dy}/(c\sigma_d) + 1, \tag{39}$$

$$\nabla_3 = 2R\left[W - F/(c\sigma_d)\right], \tag{40}$$

$$W_3 = F/(c\sigma_d), \tag{41}$$

and

$$\xi_{3,\min} = 1 - c^{-2} \approx 0.363. \tag{42}$$

It is quite obvious that there is only one solution here because, unlike (19) and (30), the right-hand side of (41) is independent of W. This performance function is closest in form to the mean-square error that is used for the LMS algorithm, so it is not unexpected that (39) should have a quadratic form with a unique minimum. By comparing W_3 to W_1 [from (19)], we see that they are equivalent when $\sigma_y = c^{-1}$. In effect, this algorithm adjusts the weights subject to the constraint that the output variance at convergence be $\sigma_y^2 = W^T R W = c^{-2}$.

D. Comparison of the Performance Functions

For ease of comparison, the performance functions have been rewritten as follows:

$$\xi_1 = 2E\left[\,|y(n)| - d_q(n)y(n)\right], \tag{43}$$

$$\xi_2 = 2E\left[1 - d_q(n)y_q(n)\right], \tag{44}$$

and

$$\xi_3 = 2E\left[(1 + y^2(n))/2 - d_q(n)y(n)\right]. \tag{45}$$

Observe that they are all similar, in that a cross-correlation between the perceptron output [i.e., $y(n)$ or $y_q(n)$] and the desired response $d_q(n)$ are being compared to either a constant (i.e., 1) or a measure of the magnitude of the output [i.e., $|y(n)|$ or $y^2(n)$]. (The constant added to $y^2(n)$ in (45) ensures that ξ_3 is nonnegative.) In this regard, they are all attempting to achieve the same goal and, as demonstrated in the previous sections, they all have a minimum point in common [given by (41)]. Therefore, in terms of finding an optimal set of weights, these performance functions can be considered to be equivalent. However, the convergence properties of the corresponding algorithms may be quite different, as demonstrated in the next section.

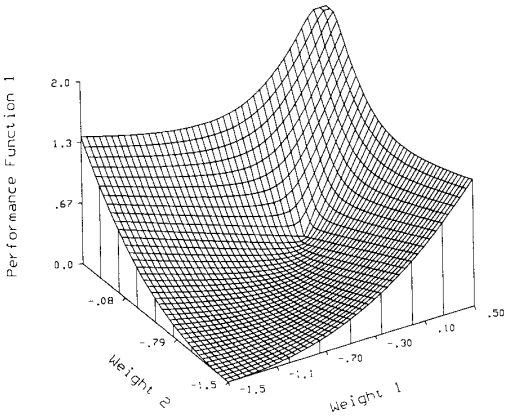

Fig. 3. Performance surface of ξ_1 truncated at 2.0.

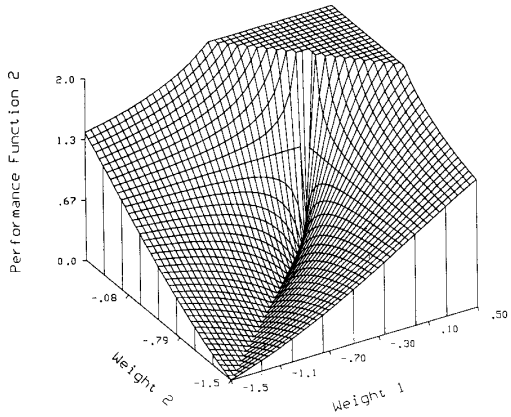

Fig. 4. Performance surface of ξ_2 truncated at 2.0.

IV. Examples of the Performance Surfaces

Some examples of the surfaces for the three performance functions are now presented. The input signals are assumed to be jointly Gaussian and $d(n)$ is generated according to (4). This allows us to use the closed-form expressions in (17), (28), and (39) for ξ_1, ξ_2, and ξ_3, respectively. Furthermore, $N = 2$, $R = I$ (the identity matrix), and $F = [-1 \quad -1]^T$, so that $\sigma_d = \sqrt{2}$ and $\rho_{dy} = F^T W = -(w_1 + w_2)$. The corresponding minimum points for ξ_1 and ξ_2 are, thus, of the form kF, where k is a *positive* scalar and the stationary point for ξ_3 is approximately 0.564 F. Figs. 3, 4, and 5 show the surfaces (truncated at 2.0) for ξ_1, ξ_2, and ξ_3, respectively. Observe that the minimum points of ξ_1 and ξ_2 are determined by the line

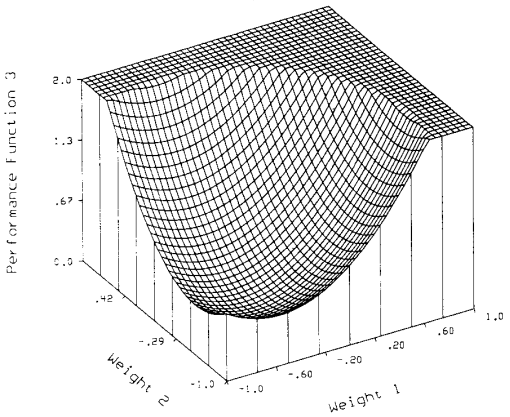

Fig. 5. Performance surface of ξ_3 truncated at 2.0.

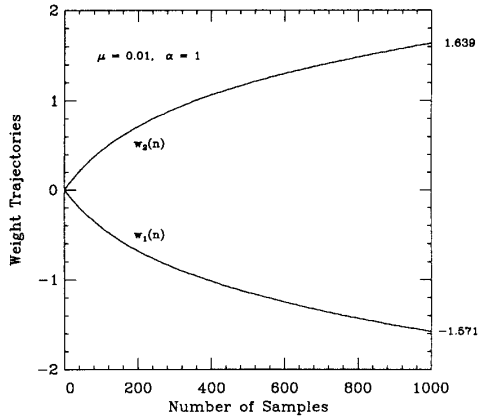

Fig. 7. Weight trajectories of the algorithm in (26).

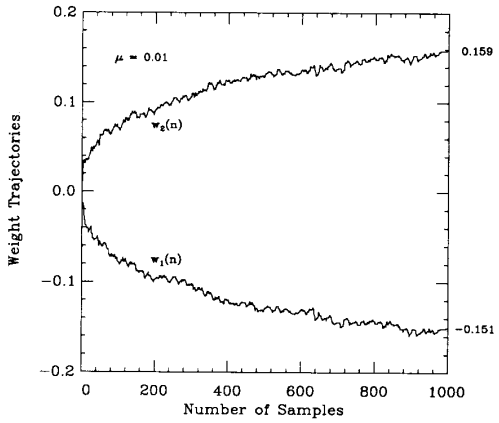

Fig. 6. Weight trajectories of the algorithm in (1).

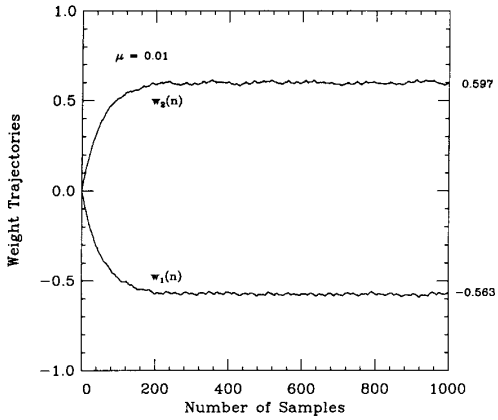

Fig. 8. Weight trajectories of the algorithm in (34).

starting at the origin and extending out such that $w_1 = w_2$ for $w_2 < 0$, as predicted, because of the elements chosen for F. The surface for ξ_3 is clearly quadratic and it has a single minimum.

The weight trajectories of the perceptron algorithms in (1), (26), and (34) are shown in Figs. 6, 7, and 8, respectively. The same conditions as above were used here, except $F = [-1 \quad 1]^T$, so that the optimal weights are the negative of each other and their trajectories can be seen more clearly. For each case, the weights were initialized to zero, the step size was $\mu = 0.01$, and the weight trajectories were averaged over 25 independent computer runs to obtain relatively smooth results. The sigmoidal nonlinearity with $\alpha = 1$ was used for the algorithm based on ξ_2. Observe that the weights converge as predicted (the weight values at iteration 1000 are shown to the right of the figures); the noisy form of the trajectories is known as misadjustment, which is an inherent property of steepest-descent algorithms [4]. [Actually, the weights in Figs. 6 and 7 have not quite converged, in the sense that the trajectories are not level as in Fig. 8. However, they are essentially optimal because $w_1(n) \approx -w_2(n)$.] Note also that the trajectories associated with ξ_2 are smoother than the others. This is a direct result of the sigmoidal nonlinearity that generates $y_q(n)$ and because the derivative used in the algorithm is relatively smooth for this particular value of α (see Fig. 2).

V. Conclusion

The performance functions of three perceptron algorithms have been presented, and analytical expressions for the minimum points based on Gaussian assumptions have been derived. It was demonstrated that the stationary points may not be unique and that there is a common minimum point for all three functions. Thus, the perceptron will have the same steady-state, decision-making characteristics regardless of which algorithm is used, but the transient properties of the learning algorithms may be quite different. One algorithm is particularly interesting because it has a unique stationary point, but the importance of this is not clear because the perceptron structure is configured such that a unique solution is not required for optimal performance.

ACKNOWLEDGMENT

The author would like to thank the anonymous Associate Editor and reviewers of this paper for their helpful suggestions.

REFERENCES

[1] F. Rosenblatt, *Principles of Neurodynamics*. New York: Spartan, 1962.

[2] B. Widrow and M. E. Hoff, Jr., "Adaptive switching circuits," *IRE WESCON Conven. Rec.*, Sept. 1960, pp. 96–104, part 4.

[3] B. Widrow, R. G. Winter, and R. A. Baxter, "Layered neural nets for pattern recognition," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 36, pp. 1109–1118, July 1988.

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

[5] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, D. E. Rumelhart and J. L. McClelland, Eds. Cambridge, MA: M.I.T. Press, 1986.

[6] R. P. Lippmann, "An introduction to computing with neutral nets," *IEEE ASSP Mag.*, vol. 4, pp. 4–22, Apr. 1987.

[7] G. O. Stone, "An analysis of the delta rule and the learning of statistical associations," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, D. E. Rumelhart and J. L. McClelland, Eds. Cambridge, MA: M.I.T. Press, 1986.

[8] J. J. Shynk and S. Roy, "Convergence properties and stationary points of a perceptron learning algorithm," *Proc. IEEE*, vol. 78, Sept. 1990.

[9] ——, "Analysis of a perceptron learning algorithm with momentum updating," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Albuquerque, NM, Apr. 1990, pp. 1377–1380.

[10] R. Price, "A useful theorem for nonlinear devices having Gaussian inputs," *IRE Trans. Inform. Theory*, vol. IT-4, pp. 69–72, June 1958.