On the System Identification Convergence Model for Perceptron Learning Algorithms

John J. Shynk

Center for Information Processing Research
Dept. of Elec. and Comp. Engineering
University of California
Santa Barbara, CA 93106

Neil J. Bershad

Dept. of Elec. and Comp. Engineering
University of California
Irvine, CA 92717

Abstract

The convergence behavior of perceptron learning algorithms has been difficult to analyze because of their inherent nonlinearity and the lack of an appropriate model for the training signals. In many cases, extensive computer simulations have been the only way of quantifying their performance. Recently we introduced a stochastic convergence model based on a system identification formulation of the training data that allows one to derive closed-form expressions for the stationary points and cost functions, as well as deterministic recursions for the transient learning behavior. In this paper, we provide an overview of this approach and describe how it is applied to single- and two-layer perceptron configurations.

1 Introduction

In recent years, multilayer perceptrons have been applied to many problems in signal processing, control, and communications (see, e.g., [1], [2], [3]). Because of the nonlinear and time-varying nature of the learning algorithms, their convergence properties are not well understood and it is difficult to predict their behavior in a given application. The perceptron parameters (i.e., number of nodes, algorithm step size, etc.) are usually chosen in an ad hoc manner, and as a result the network generally yields suboptimal performance. Thus, it would be useful to have a model of convergence that provides information about the average behavior of a perceptron for a specific scenario which might also be useful for other situations.

Towards this end, we recently introduced a stochastic convergence model for a single-layer perceptron based on a *system identification* formulation [4], [5]. We examined the *transient* and *steady-state* convergence behavior of Rosenblatt's algorithm [5], [6], and the backpropagation (BP) algorithm [7], [8], using a Gaussian data model. These results were extended to a partially adaptive two-layer network [9] and to the case of noisy input patterns [6]. Nonseparable data models for a single-layer perceptron were also considered [10], [11].

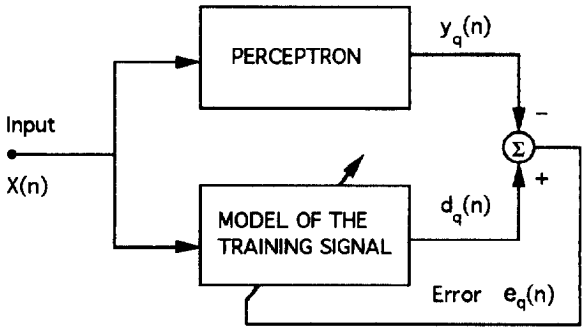

Figure 1: System identification model.

In this paper, we present a summary of the system identification model and provide examples of its use for single- and two-layer perceptrons. Expressions are given for the transient weight behavior, stationary points, and a performance function. Computer simulations are included to demonstrate the accuracy of the analytical results.

2 System Identification Model

A block diagram of the system identification model is shown in Figure 1. Observe that a nonlinear model representing the training data is in parallel with the neural network (perceptron). Both have the same input $X(n) = [x_1(n), \ldots, x_N(n)]^T$, which we assume is a zero-mean Gaussian vector with independent and

1058-6393/95 $4.00 © 1995 IEEE

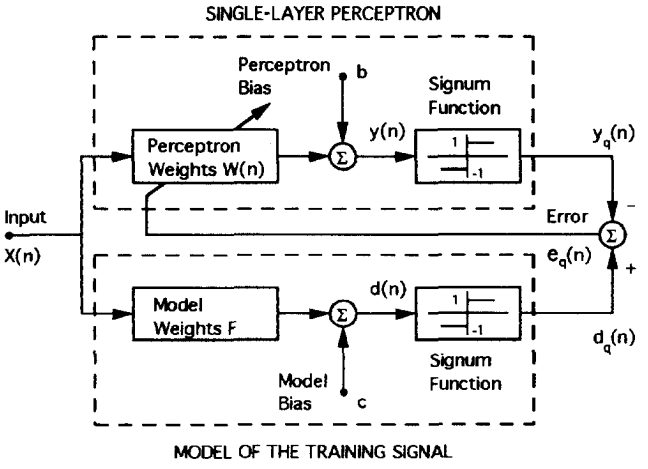

Figure 2: System identification model for a single-layer perceptron.

identically distributed components. Thus, its covariance matrix is $R_x = E[X(n)X^T(n)] = \sigma_x^2 I$. In our analysis of convergence, we evaluate expectations of *nonlinear* functionals. This Gaussian model allows us to obtain closed-form expressions that otherwise would be difficult or impossible to derive. In general, R_x may be a full matrix; however, for convenience, we will assume that it is diagonal as described above.

Figure 2 shows a specific example of a single-layer perceptron with adaptive weights $W(n) = [w_1(n), \ldots, w_N(n)]^T$, fixed bias b, and output $y(n) = W^T(n)X(n)$ [5]. In order to proceed with an analysis of this *nonlinear adaptive filter*, it is necessary that a specific model be chosen for the training signal $d_q(n)$. Obviously the particular application determines how $d_q(n)$ is generated. The model in Figure 2 is similar in form to a single-layer perceptron having fixed weights $F = [f_1, \ldots, f_N]^T$ and fixed bias c. This configuration corresponds to a *separable* data model, such that the single-layer perceptron weights may be adjusted to drive the error $e_q(n) = d_q(n) - y_q(n)$ to zero. An example of a *nonseparable* data model [10] for a single-layer perceptron is shown in Figure 3. A partially adaptive two-layer perceptron and a possible data model are illustrated in Figure 4 [9].

Clearly, the accuracy of the convergence results will depend on the model chosen for $d_q(n)$. The separable and nonseparable data models (Figures 2 and 3) are two possible representations that may only approximate the "true" underlying model in a given application. Nevertheless, the insight provided by an analysis of these systems may be useful in understanding their transient and steady-state convergence behavior in applications where the mechanism generating $d_q(n)$ is unknown or is not easily modeled.

In our early work, we examined the steady-state and transient convergence behavior of the single-layer separable model in Figure 2. Both Rosenblatt's algorithm [1], [12], and the backpropagation (BP) algorithm [13], [14], were considered. Rosenblatt's algorithm has a weight update similar to the least-mean-square (LMS) algorithm [15], except that the error term is based on the nonlinear perceptron output:[1]

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

where $\mu > 0$ is the step size, and the nonlinearity corresponds to the signum function. The BP algorithm for a single-layer perceptron is given by[2]

$$W(n+1) = W(n) + 2\mu X(n)[d_q(n) - y_q(n)]y_q'(n) \quad (2)$$

where $y_q'(n)$ is the ordinary derivative of the nonlinear output; thus, the nonlinearity must be a smooth function such as the hyperbolic tangent.

Because $X(n)$ is Gaussian, we may employ Bussgang's theorem [17], Price's theorem [18], and a modification by Pawula [19] to derive closed-form expressions for the various expectations arising from an analysis of (1) and (2). For example, stationary points of (1) are determined from the orthogonality condition [15]:

$$E[X(n)[d_q(n) - y_q(n)]] = \underline{0} \quad (3)$$

by solving for the weight vector $W \triangleq \lim_{n\to\infty} W(n)$. Using these theorems and conditioning on W, we may write expectations of nonlinear functionals in terms of the corresponding linear expressions based on the signals before the nonlinearities in Figure 2. For example,

$$E[X(n)d_q(n)] = (\sigma_x^2/s\sigma_d)e^{-c^2/2\sigma_d^2}F \quad (4)$$

where $\sigma_d^2 = E[d^2(n)] = F^T F\sigma_x^2$ is the variance (power) of the signal before the nonlinearity, and $s = \sqrt{\pi/2}$ is a constant that arises because of the Gaussian model.

[1]The error term in the LMS algorithm is derived from the perceptron output before the nonlinear device (signum function) in Figure 2.

[2]Although the BP algorithm was originally designed to update the weights of a multilayer perceptron, it is possible to derive a single-layer version as given in (2). The resulting algorithm is similar to the LMS algorithm because it is a gradient technique that attempts to minimize the mean-square error; on the other hand, Rosenblatt's algorithm is based on a different performance function (see [16]) when viewed as a gradient algorithm.

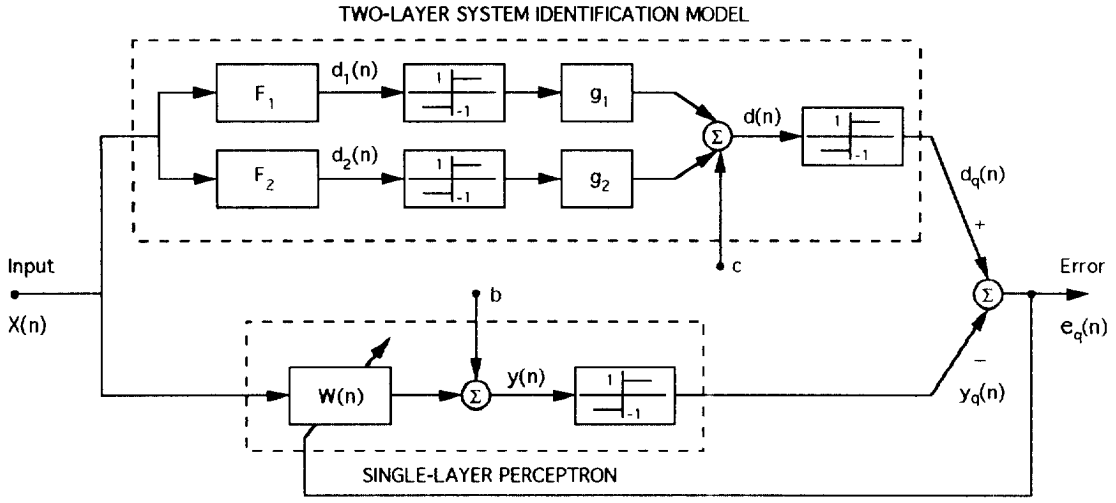

Figure 3: Nonseparable data model for a single-layer perceptron.

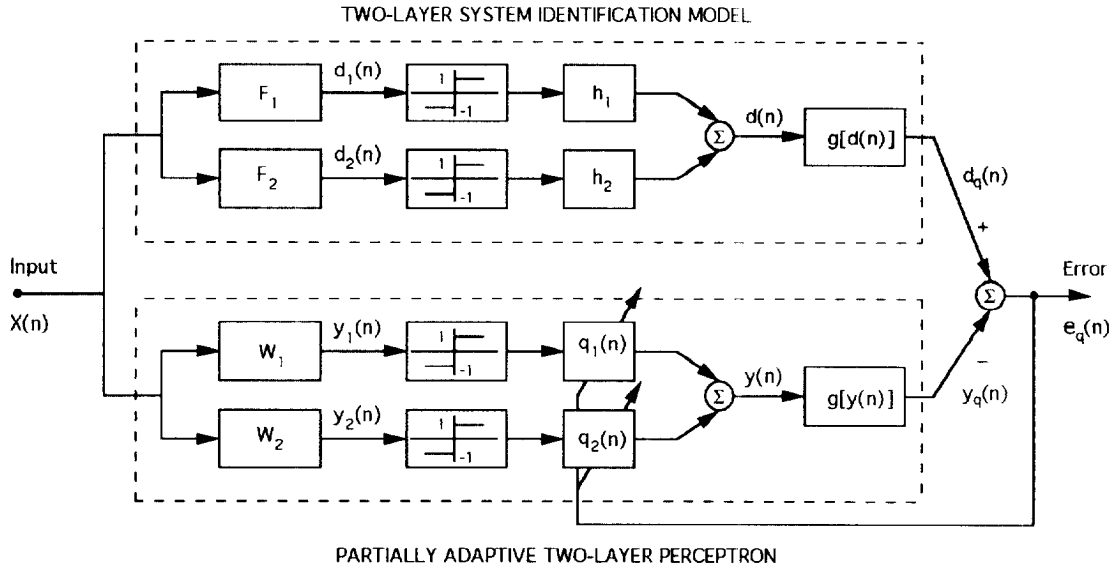

Figure 4: System identification model for a two-layer perceptron.

Using this approach, we have derived expressions for the stationary points and the performance function for Rosenblatt's algorithm in (1) for single-layer separable (Figure 2) and nonseparable (Figure 3) models. Some examples of these results are given in Section 3. Similar results have been derived for the single-layer BP algorithm in (2), but only for the case of zero bias terms and a separable data model. This is because the previously mentioned theorems are not easily evaluated for nonzero bias terms nor for more complicated (e.g., nonseparable) models of $d_q(n)$. Expressions for the probability of correct classification have also been derived but will not be discussed here (see [8] and [20]).

This approach has been extended to a partially adaptive two-layer perceptron where only the output-layer weights are adjusted. We have derived equations that determine the stationary points of the output-layer weights; interestingly, the results also specify conditions on the hidden-layer weights in order that they be stationary points (in the event that they were adapted). Section 4 presents some examples of our results for this case.

Finally, we present in Section 5 some recent work describing the convergence behavior of the model in Figure 2 when the input of the perceptron is corrupted by noise. This noise influences the stationary points and transient behavior of Rosenblatt's learning algorithm.

3 Convergence Example for a Single-Layer Perceptron

Observe for the nonseparable data model in Figure 3 that $X(n)$ is processed in parallel by two sets of weight vectors, $F_1 = [f_{11}, \ldots, f_{1N}]^T$ and $F_2 = [f_{21}, \ldots, f_{2N}]^T$, whose outputs are quantized independently by signum function nonlinearities. These binary signals are then weighted by g_1 and g_2, added together, and quantized by a third signum function to yield the training signal $d_q(n)$. Thus, the model has the form of a two-layer perceptron with *fixed* weights, and with two nodes in the hidden layer and a single output node. In order to simplify our analysis, the hidden layer does not contain any bias terms (see [11] for a discussion of this case). Note, however, that the problem is still a nontrivial one because of the bias term c. By appropriately choosing $\{F_i\}$, $\{g_i\}$, and c, the training signal can be designed such that it corresponds to a nonseparable data model.

For example, consider the first quadrant of the (g_1, g_2) plane where $g_1 > 0$ and $g_2 > 0$. The results described below are similar for the other three quadrants. By varying g_1 and g_2, the (d_1, d_2) signal space can be partitioned by the two-layer data model in only four ways as follows: Case 1: $g_1 - g_2 > |c|$, Case 2: $g_1 - g_2 < -|c|$, Case 3: $(-|c| < g_1 - g_2 < |c|) \cap (g_1 + g_2 > |c|)$, and Case 4: $g_1 + g_2 < |c|$. For Cases 1 and 2, only one arm of the two-layer data model dominates so that either F_1 (Case 1) or F_2 (Case 2) alone determines the output $d_q(n)$. This occurs because the outputs of the hidden layer are binary and g_1 is sufficiently larger than g_2 (Case 1), or vice versa (Case 2), to control the signum function. The signal space is clearly *separable* by a hyperplane through the origin. On the other hand, the data model is *nonseparable* for Case 3, and both F_1 and F_2 together influence the training signal. The relative values of g_1 and g_2 are such that neither arm dominates the output $d_q(n)$. In Case 4, g_1 and g_2 are sufficiently small compared to c so that $d_q(n)$ is determined only by the sign of c. Consequently, the signal space has only one region and the resulting classification problem is trivial.

The stationary points of Rosenblatt's algorithm for the four cases above are given by (for the first quadrant) [10]

Case 1 $(g_1 - g_2 > |c|)$:

$$W = kF_{1n}, \text{ for any } k > 0 \text{ and } b = 0 \quad (\text{or } k = \infty \text{ for } b \neq 0). \tag{5}$$

Case 2 $(g_1 - g_2 < -|c|)$:

$$W = kF_{2n}, \text{ for any } k > 0 \text{ and } b = 0 \quad (\text{or } k = \infty \text{ for } b \neq 0). \tag{6}$$

Case 3 $((-|c| < g_1 - g_2 < |c|) \cap (g_1 + g_2 > |c|))$:

$$W = k(F_{1n} + F_{2n}), \tag{7}$$
$$\text{for } k = |b|/[\sigma_x \sqrt{2(1+\rho_{12})} \log(2/(1+\rho_{12}))].$$

Case 4 $(g_1 + g_2 < |c|)$:

$$W = \underline{0}, \text{ for any } b. \tag{8}$$

Note that the model weight vectors are normalized, i.e., $F_{in} = F_i/\sqrt{F_i^T F_i}$ $(i = 1, 2)$; the cross-correlation coefficient is given by $\rho_{12} = F_{1n}^T F_{2n}$. The stationary points corresponding to the other quadrants of the (g_1, g_2) plane are determined in a similar manner, and they differ from (5)–(8) only in the signs before F_{1n} and F_{2n}; these are summarized in Figure 5. Overall there are nine different regions depending on the relative values of g_1, g_2, and $|c|$.

If we view the update in (1) as a gradient-descent method, then its performance function is [16]

$$\xi = 2E[|y(n)| - d_q(n)y(n)]. \tag{9}$$

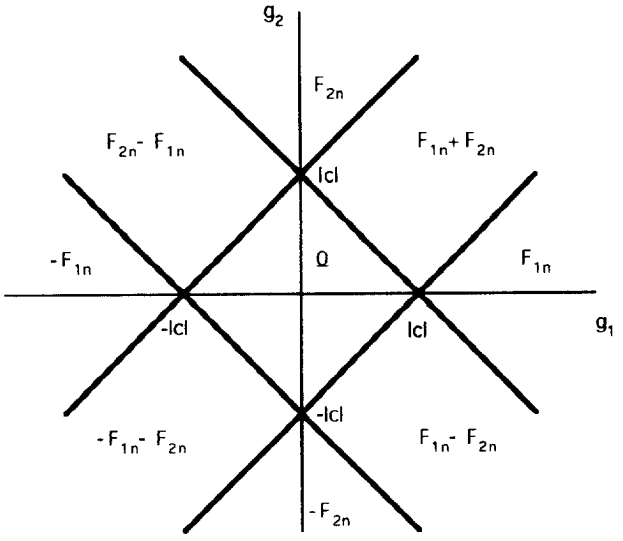

Figure 5: Stationary points for the nonseparable data model.

The update term in (1) is obtained by differentiating with respect to $W(n)$ an instantaneous estimate of (9), obtained by ignoring the expectation operator $E[\cdot]$. Evaluating the expectations in (9), it can be shown that

$$\begin{aligned}\xi &= 2(\sigma_y/s)e^{-b^2/2\sigma_y^2} + 2b\mathrm{erf}(b/\sqrt{2}\sigma_y) \\ &\quad -(b/\pi)\sin^{-1}(\rho_{12}) \\ &\quad \times(\mathrm{sgn}(g_1+g_2+c) - \mathrm{sgn}(g_1+g_2-c) \\ &\quad -\mathrm{sgn}(g_1-g_2-c) + \mathrm{sgn}(g_1-g_2+c)) \\ &\quad -(\sigma_x/2s)(W^T F_{1n} + W^T F_{2n}) \\ &\quad \times(\mathrm{sgn}(g_1+g_2+c) + \mathrm{sgn}(g_1+g_2-c)) \\ &\quad -(\sigma_x/2s)(W^T F_{1n} - W^T F_{2n}) \\ &\quad \times(\mathrm{sgn}(g_1-g_2+c) + \mathrm{sgn}(g_1-g_2-c)) \quad (10)\end{aligned}$$

where $\mathrm{erf}(\cdot)$ is the "standard" error function:

$$\mathrm{erf}(a) = \frac{2}{\sqrt{\pi}}\int_0^a e^{-z^2}dz. \quad (11)$$

By varying W for $N = 2$ and with fixed parameters in the data model, it is possible to plot performance surfaces for the algorithm using this expression.

The weight trajectories of Rosenblatt's algorithm for Case 3 (the nonseparable configuration) are shown in Figure 6. There are two adaptive weights and the input signals are independent Gaussian sequences with zero means and $\sigma_x^2 = 1$. The underlying model vectors are $F_1 = [1,1]^T$ and $F_2 = [1,-1]^T$ so that

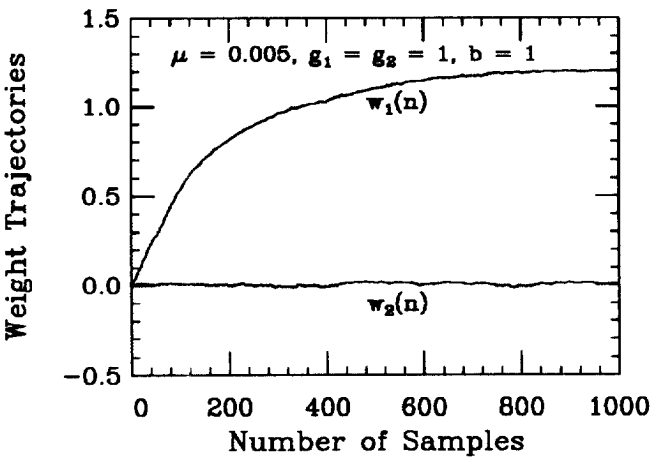

Figure 6: Learning curves for the nonseparable data model.

$F_1^T F_1 = F_2^T F_2 = 2$ and $\rho_{12} = 0$. The bias terms are $b = c = 1$. The step size is $\mu = 0.005$, the weights are initialized to zero, and the weight trajectories are averaged over 25 independent computer runs to generate smooth trajectories.

Observe that the analytical results accurately predict the converged weight values, given by $k(F_{1n}+F_{2n})$ with $k = 0.849$ (from (7)). Since $F_1^T F_1 = F_2^T F_2 = 2$, the weights should converge to $W = \sqrt{2}k[1,0]^T \approx [1.201, 0]^T$, and this is demonstrated in the figure. The corresponding performance surface (truncated at 2.0) is shown in Figure 7 (where instead $F_1 = [-1,-1]^T$ and $F_2 = [-1,1]^T$ for ease of viewing). Note that the global minimum of the performance function is at the point $(-1.201, 0)$ (corresponding to $k(F_{1n}+F_{2n})$), which is consistent with the learning curves in Figure 6.

4 Convergence Example for a Two-Layer Perceptron

The training signal of the two-layer model in Figure 4 is given by (with $g[\cdot] = \mathrm{sgn}[\cdot]$)

$$d_q(n) = \mathrm{sgn}[h_1\mathrm{sgn}(d_1(n)) + h_2\mathrm{sgn}(d_2(n))] \quad (12)$$

where $\{d_i(n)\}$ are intermediate training signals computed as $d_i(n) = F_i^T X(n)$, and $\{F_i\}$ are fixed weight vectors analogous to those of the nonseparable data model in Figure 3. The perceptron output is similarly computed as

$$y_q(n) = \mathrm{sgn}[q_1(n)\mathrm{sgn}(W_1^T X(n)) + q_2(n)\mathrm{sgn}(W_2^T X(n))]. \quad (13)$$

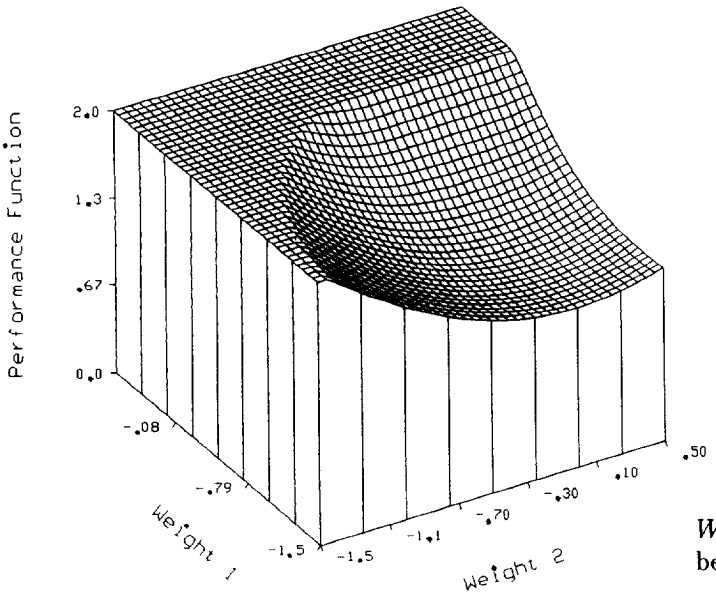

Figure 7: Performance surface for the nonseparable data model.

Observe that only the outer-layer weights $\{q_i(n)\}$ are time-varying; they are adjusted by Rosenblatt's algorithm as follows:

$$q_i(n+1) = q_i(n) + 2\mu e_q(n)\text{sgn}[W_i^T(n)X(n)]. \quad (14)$$

The stationary points of (14) are determined by

$$E[d_q(n)\text{sgn}[W_i^T(n)X(n)]] = E[y_q(n)\text{sgn}[W_i^T(n)X(n)]] \quad (15)$$

for $i = 1, 2$. It is clear that $e_q(n) = 0$ when the parameters of the two-layer perceptron are equal to those of the nonlinear system, i.e., $W_i = F_i$ and $q_i = h_i$. Furthermore, because the signum function is amplitude independent, the error is also zero when $W_i = k_i F_i$ where $k_i > 0$, analogous to Cases 1 and 2 in (5)–(6). Both of these solutions satisfy the orthogonality condition in (15).

However, there are additional stationary points that satisfy (15) but do not yield exactly zero error. It can be shown that these weights are determined by [9]

$$\angle(F_1, W_1) + \angle(F_1, W_2) = \angle(W_1, W_2) \quad (16)$$

$$\angle(F_2, W_1) + \angle(F_2, W_2) = \angle(W_1, W_2) \quad (17)$$

where $\angle(A, B)$ is the principal value of $\cos^{-1}(A^T B / \sqrt{A^T A B^T B})$. This result corresponds to $q_1 = q_2 > 0$, which is the only case of interest (a similar result can be derived for $q_1 = q_2 < 0$); the case of $q_1 \neq q_2$ can be handled by a single-layer perceptron.

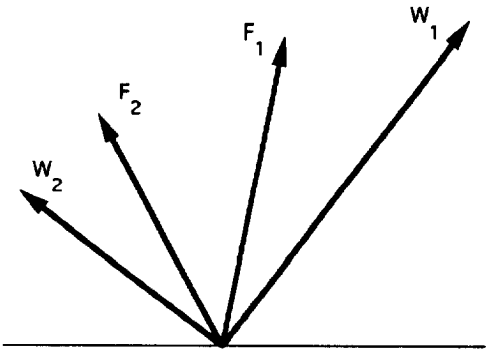

Figure 8: Example solution of (16) and (17).

Equation (16) requires that F_1 lie between W_1 and W_2 on the same plane, as does (17) for F_2. It can also be shown that [9]

$$\angle(F_1, W_1) = \angle(F_2, W_2) \quad (18)$$

$$\angle(F_1, W_2) = \angle(F_2, W_1). \quad (19)$$

An example of this result is shown in Figure 8 where all four vectors lie on the same plane. The data vector $X(n)$ projects onto this plane, and the perpendiculars to the vectors define the regions $F_i^T X(n) > 0$ and $W_i^T X(n)$, for $i = 1, 2$. Note that (16)–(19) specify conditions on the hidden-layer weights $\{W_i\}$ even though only the output-layer weights $\{q_i(n)\}$ were adapted by Rosenblatt's algorithm.

An example of the learning curves for the output layer of the two-layer perceptron is shown in Figure 9. These trajectories were obtained by averaging the weights of Rosenblatt's algorithm over 50 independent computer runs. The input variance is $\sigma_x^2 = 1$, the algorithm step size is $\mu = 0.05$, and the initial weight values are $q_1(0) = -1$ and $q_2(0) = 1.5$. The model weights are $F_1 = [1, 1]^T$, $F_2 = [1, -1]^T$, and $h_1 = h_2 = 1$, and the perceptron hidden-layer weights are $W_1 = [1, 2]^T$ and $W_2 = [1, -2]^T$. Note that we have chosen the perceptron weights such that the conditions in (16) and (17) are satisfied. Since $h_1 = h_2 > 0$, the weights in the perceptron output layer should converge such that $q_1 = q_2 > 0$ on average. Observe in Figure 9 that the weights converge essentially to the same value (≈ 0.25), confirming that stationary points occur not only when $W_i = k_i F_i$ (for $i = 1, 2$ and positive k_i) but also when the angle relationships in (16) and (17) are satisfied by the hidden-layer weights.

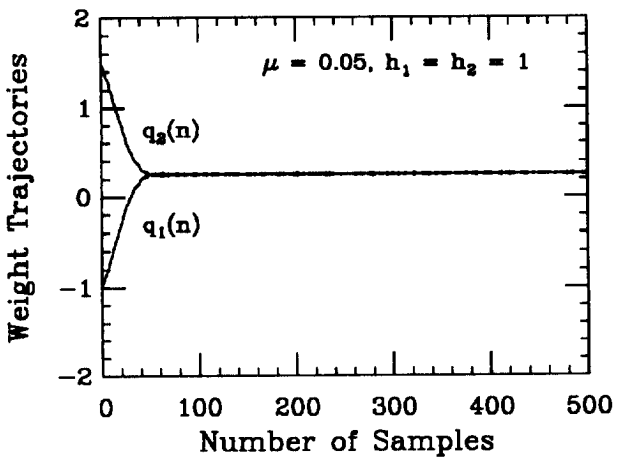

Figure 9: Learning curves for the two-layer data model.

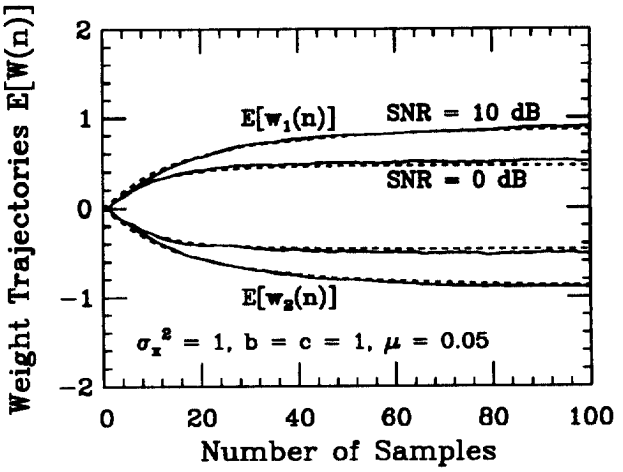

Figure 10: Transient weight behavior for the single-layer data model with noisy input.

5 Convergence Example for a Single-Layer Perceptron with Noisy Input

Assume that the input of the perceptron (but *not* the model) in Figure 2 is replaced by $\tilde{X}(n) = X(n) + V(n)$ where $V(n)$ is a white Gaussian noise vector with zero mean and covariance $\sigma_v^2 I$. The corresponding signal-to-noise ratio is SNR $= \sigma_x^2/\sigma_v^2$. Conditioning on $W(n)$ yields [6]

$$E[W(n+1)|W(n)] = W(n) + 2\mu E[e_q(n)\tilde{X}(n)|W(n)]. \quad (20)$$

Recall that $d_q(n) = \text{sgn}(F^T X(n) + c)$, while $y_q(n) = \text{sgn}(W^T(n)\tilde{X}(n) + b)$. Using methods similar to those employed for the noiseless case (see [6]), we can write

$$\begin{aligned} E[d_q(n)\tilde{X}(n)|W(n)] &= E[d_q(n)X(n)|W(n)] \\ &\quad + E[d_q(n)V(n)|W(n)] \\ &= E[d_q(n)X(n)|W(n)], \end{aligned} \quad (21)$$

which simplifies because $X(n)$ and $V(n)$ are assumed to be uncorrelated. Thus (21) reduces to the same result in (4). The evaluation of $E[y_q(n)\tilde{X}(n)|W(n)]$ is similar to that of the noiseless case [6], except that σ_x^2 is replaced by $\sigma_x^2 + \sigma_v^2 = \sigma_{\tilde{x}}^2$, i.e.,

$$\begin{aligned} E[y_q(n)\tilde{X}(n)|W(n)] &= [\sigma_{\tilde{x}}/(s(W^T(n)W(n))^{1/2})] \\ &\quad \times e^{-b^2/(2\sigma_{\tilde{x}}^2 W^T(n)W(n))} W(n). \end{aligned} \quad (22)$$

Thus, the average weight update is given by

$$\begin{aligned} E[W(n+1)] &= E[W(n)] + (2\mu/s) \\ &\quad \times \Big[(\sigma_x e^{-c^2/(2\sigma_d^2)}/(F^T F)^{1/2})F \\ &\quad - (\sigma_{\tilde{x}} e^{-b^2/(2\sigma_{\tilde{x}}^2 \gamma(n))}/\gamma^{1/2}(n))E[W(n)]\Big]. \end{aligned} \quad (23)$$

To continue, we need a recursion for $\gamma(n) = E[W^T(n)W(n)]$, as follows:

$$\begin{aligned} \gamma(n+1) &= \gamma(n) + (4\mu/s) \\ &\quad \times \Big[\sigma_x (e^{-c^2/(2\sigma_d^2)}/(F^T F)^{1/2}) E[W^T(n)]F \\ &\quad - \sigma_{\tilde{x}} e^{-b^2/(2\sigma_{\tilde{x}}^2 \gamma(n))} \gamma^{1/2}(n)\Big] \\ &\quad + 4\mu^2 N \sigma_{\tilde{x}}^2 [2 - (4/\pi)\sin^{-1}\tilde{\rho}(n)] \end{aligned} \quad (24)$$

where $\tilde{\rho}(n) = \sigma_x F^T E[W(n)]/(\sigma_{\tilde{x}}^2 F^T F \gamma(n))^{1/2}$. The corresponding convergence point is

$$W = \Big[\mu \sigma_{\tilde{x}}^2 N s [2 - (4/\pi)\sin^{-1}(\tilde{\rho})] e^{-c^2/(2\sigma_d^2)}/\tilde{\Delta}\Big] F \quad (25)$$

where $\tilde{\Delta} = (F^T F)^{1/2} |\sigma_{\tilde{x}}^2 e^{-b^2/(\sigma_{\tilde{x}}^2 \gamma)} - \sigma_x^2 e^{-c^2/\sigma_d^2}|/\sigma_x$.

For the simulation in Figure 10, $F = [1, -1]^T$ and $c = 1$ for the underlying data model generating $d_q(n)$. The bias term of the perceptron is fixed at $b = 1$, while the step size $\mu = 0.05$ and the input variance $\sigma_x^2 = 1$. The weight trajectories were averaged over 500 independent computer runs. Two signal-to-noise ratios were considered: SNR = 0 dB and 10 dB. Observe that the stationary points are influenced by the noise power, as predicted by the expression in (25). The analytical model (dotted lines) accurately predicts the behavior observed in the simulations (solid lines). We should mention, however, that this accuracy deteriorates for very low SNRs (< -10 dB), where the approximations used in the analysis may not be very accurate.

6 Concluding Remarks

We have presented an overview of our work on the convergence analysis of perceptron learning algorithms. A stochastic Gaussian model is employed in a system identification formulation to derive closed-form expressions for the transient (mean) weight behavior, stationary points, performance functions, and probability of correct classification. The Gaussian model allows us to use Price's theorem and its modification by Pawula to evaluate expectations of nonlinear functionals. Although the system identification approach is quite general, it may be difficult to apply these techniques to more complicated scenarios such as three-layer perceptron networks. Furthermore, the underlying model for the training data may not be representative of some real-world applications involving, for example, speech and image signals. Nevertheless, we believe this approach can provide greater insight into the behavior of these nonlinear adaptive systems, complementing that obtained by computer simulation studies.

References

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

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

[3] B. Widrow and M. A. Lehr, "30 years of adaptive neural networks: Perceptron, madaline, and backpropagation," *Proc. IEEE*, vol. 78, pp. 1415–1441, Sept. 1990.

[4] J. J. Shynk and S. Roy, "Convergence properties and stationary points of a perceptron learning algorithm," *Proc. IEEE*, vol. 78, pp. 1599–1604, Oct. 1990.

[5] J. J. Shynk and N. J. Bershad, "Steady-state analysis of a single-layer perceptron based on a system identification model with bias terms," *IEEE Trans. Circuits and Systems*, vol. 38, pp. 1030–1042, Sept. 1991.

[6] S. N. Diggavi, J. J. Shynk, and N. J. Bershad, "Convergence models for Rosenblatt's perceptron learning algorithm," *IEEE Trans. Signal Processing*, to be published.

[7] N. J. Bershad, J. J. Shynk, and P. L. Feintuch, "Statistical analysis of the single-layer backpropagation algorithm: Part I – Mean weight behavior," *IEEE Trans. Signal Processing*, vol. 41, pp. 573–582, Feb. 1993.

[8] N. J. Bershad, J. J. Shynk, and P. L. Feintuch, "Statistical analysis of the single-layer backpropagation algorithm: Part II – MSE and classification performance," *IEEE Trans. Signal Processing*, vol. 41, pp. 583–591, Feb. 1993.

[9] N. J. Bershad, J. J. Shynk, J. L. Vaughn, and C. F. N. Cowan, "Stochastic convergence analysis of a partially adaptive two-layer perceptron using a system identification model," *Signal Processing*, to be published.

[10] J. J. Shynk and N. J. Bershad, "Stationary points of a single-layer perceptron for nonseparable data models," *Neural Networks*, vol. 6, pp. 189–202, Mar.-Apr. 1993.

[11] N. J. Bershad and J. J. Shynk, "Performance analysis of a converged single-layer perceptron for nonseparable data models with bias terms," *IEEE Trans. Signal Processing*, vol. 42, pp. 175–188, Jan. 1994.

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

[13] 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: MIT Press, 1986.

[14] P. J. Werbos, "Backpropagation: Past and future," in *Proc. IEEE Int. Conf. Neural Networks*, (San Diego, CA), pp. I–343–353, July 1988.

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

[16] J. J. Shynk, "Performance surfaces of a single-layer perceptron," *IEEE Trans. Neural Networks*, vol. 1, pp. 268–274, Sept. 1990.

[17] J. J. Bussgang, "Cross-correlation functions of amplitude-distorted Gaussian signals," Tech. Rep. 216, Research Laboratory of Electronics, MIT, Cambridge, MA, Mar. 1952.

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

[19] R. F. Pawula, "A modified version of Price's theorem," *IEEE Trans. Information Theory*, vol. IT-13, pp. 285–288, Apr. 1967.

[20] I. Engel and N. J. Bershad, "A transient learning comparison of Rosenblatt, backpropagation, and LMS algorithms for a single-layer perceptron for system identification," *IEEE Trans. Signal Processing*, vol. 42, pp. 1247–1251, May 1994.