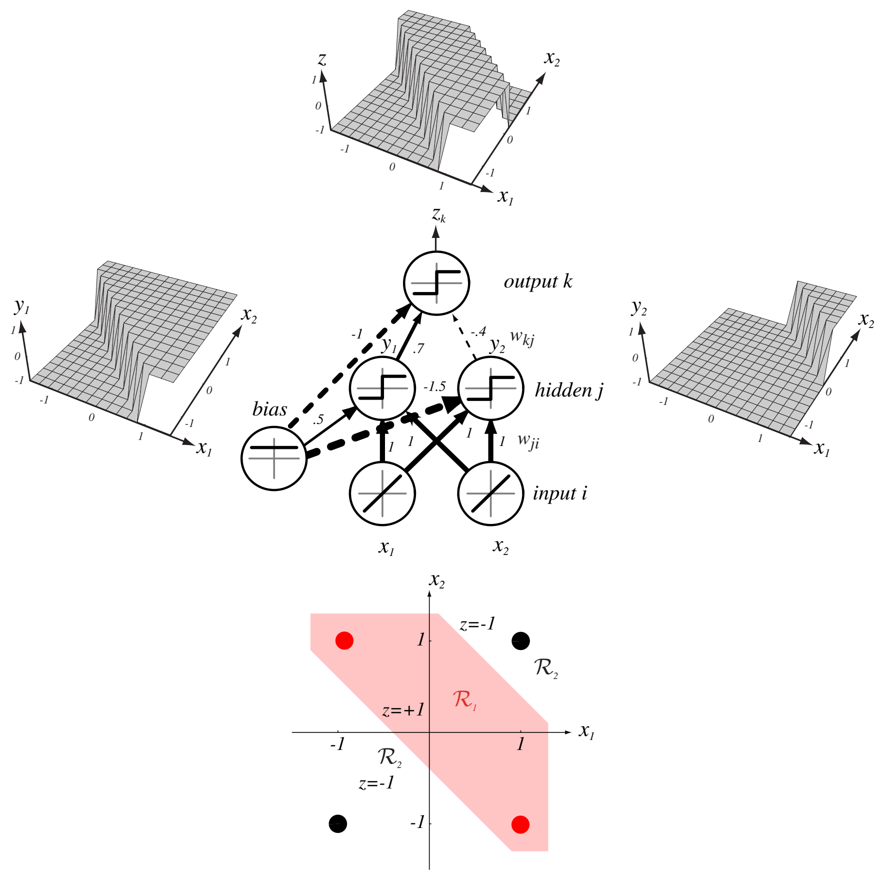

FIGURE 6.1. The two-bit parity or exclusive-OR problem can be solved by a three-layer network. At the bottom is the two-dimensional feature x_1x_2-space, along with the four patterns to be classified. The three-layer network is shown in the middle. The input units are linear and merely distribute their feature values through multiplicative weights to the hidden units. The hidden and output units here are linear threshold units, each of which forms the linear sum of its inputs times their associated weight to yield *net*, and emits a +1 if this *net* is greater than or equal to 0, and −1 otherwise, as shown by the graphs. Positive or "excitatory" weights are denoted by solid lines, negative or "inhibitory" weights by dashed lines; each weight magnitude is indicated by the line's thickness, and is labeled. The single output unit sums the weighted signals from the hidden units and bias to form its *net*, and emits a +1 if its *net* is greater than or equal to 0 and emits a −1 otherwise. Within each unit we show a graph of its input-output or activation function—$f(net)$ versus *net*. This function is linear for the input units, a constant for the bias, and a step or sign function elsewhere. We say that this network has a 2-2-1 fully connected topology, describing the number of units (other than the bias) in successive layers. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

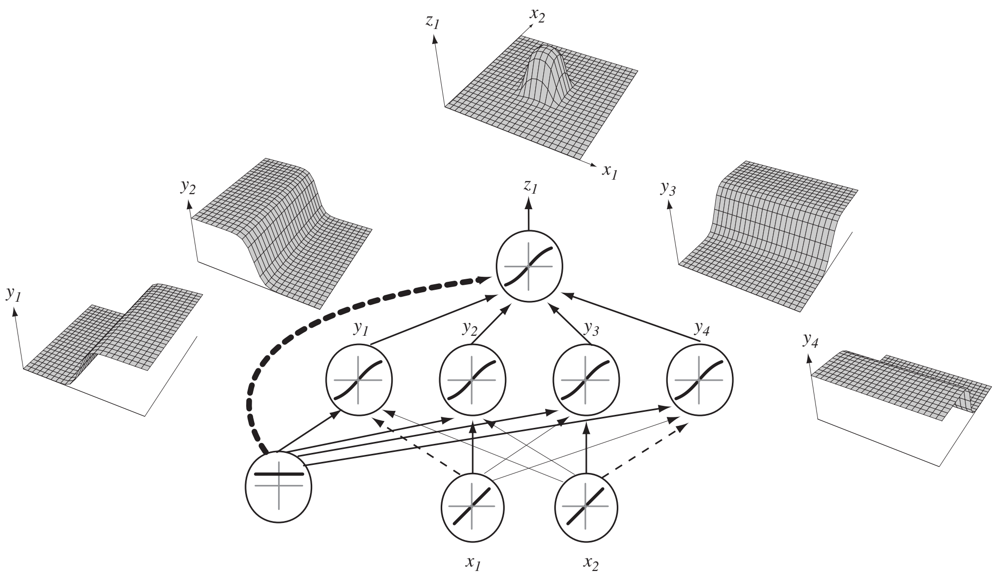

FIGURE 6.2. A 2-4-1 network (with bias) along with the response functions at different units; each hidden output unit has sigmoidal activation function $f(\cdot)$. In the case shown, the hidden unit outputs are paired in opposition thereby producing a "bump" at the output unit. Given a sufficiently large number of hidden units, any continuous function from input to output can be approximated arbitrarily well by such a network. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

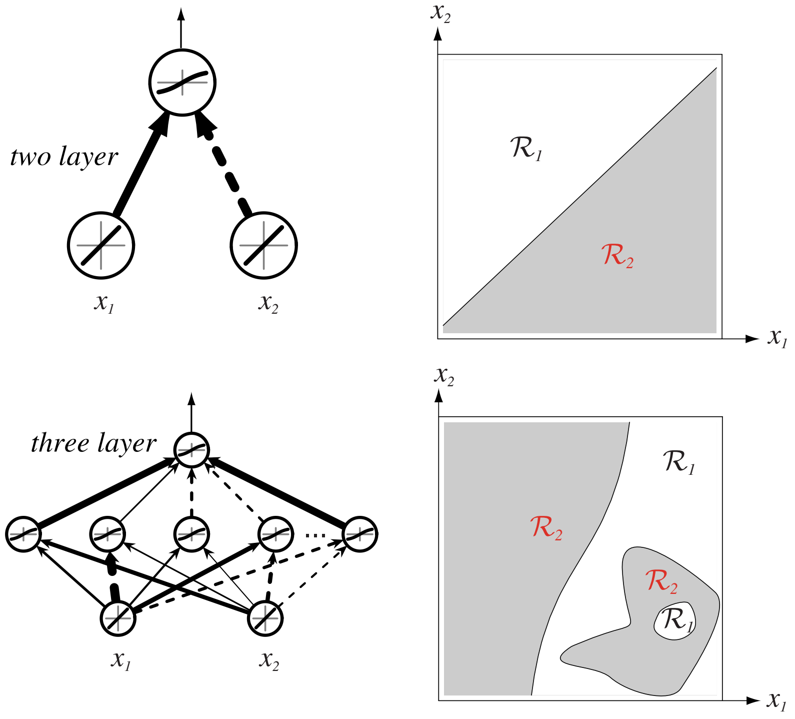

FIGURE 6.3. Whereas a two-layer network classifier can only implement a linear decision boundary, given an adequate number of hidden units, three-, four- and higher-layer networks can implement arbitrary decision boundaries. The decision regions need not be convex or simply connected. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

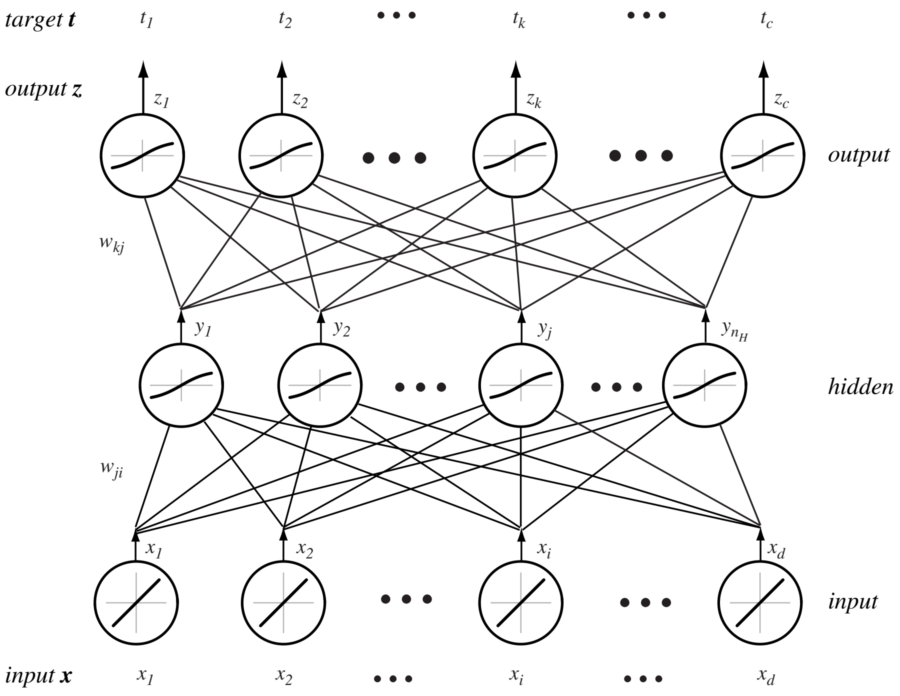

FIGURE 6.4. A d-n_H-c fully connected three-layer network and the notation we shall use. During feedforward operation, a d-dimensional input pattern $\mathbf{x}$ is presented to the input layer; each input unit then emits its corresponding component x_i. Each of the n_H hidden units computes its net activation, net_j, as the inner product of the input layer signals with weights w_{ji} at the hidden unit. The hidden unit emits $y_j = f(net_j)$, where $f(\cdot)$ is the nonlinear activation function, shown here as a sigmoid. Each of the c output units functions in the same manner as the hidden units do, computing net_k as the inner product of the hidden unit signals and weights at the output unit. The final signals emitted by the network, $z_k = f(net_k)$, are used as discriminant functions for classification. During network training, these output signals are compared with a teaching or target vector $\mathbf{t}$, and any difference is used in training the weights throughout the network. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

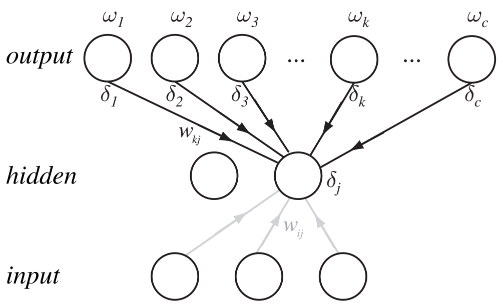

FIGURE 6.5. The sensitivity at a hidden unit is proportional to the weighted sum of the sensitivities at the output units: $\delta_j = f'(net_j) \sum_{k=1}^{c} w_{kj}\delta_k$. The output unit sensitivities are thus propagated "back" to the hidden units. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

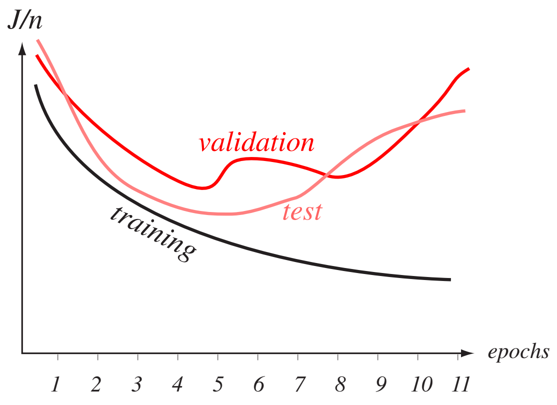

FIGURE 6.6. A learning curve shows the criterion function as a function of the amount of training, typically indicated by the number of epochs or presentations of the full training set. We plot the average error per pattern, that is, $1/n \sum_{p=1}^{n} J_p$. The validation error and the test or generalization error per pattern are virtually always higher than the training error. In some protocols, training is stopped at the first minimum of the validation set. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

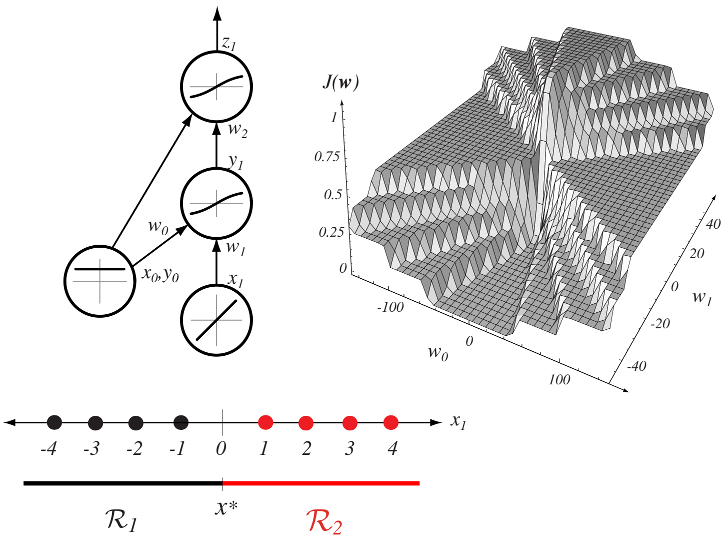

FIGURE 6.7. Eight one-dimensional patterns (four in each of two classes) are to be learned by a 1-1-1 network with steep sigmoidal hidden and output units with bias. The error surface as a function of w_0 and w_1 is also shown, where the bias weights are assigned their final values. The network starts with random weights; through stochastic training, it descends to a global minimum in error. Note especially that a low error solution exists, which indeed leads to a decision boundary separating the training points into their two categories. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

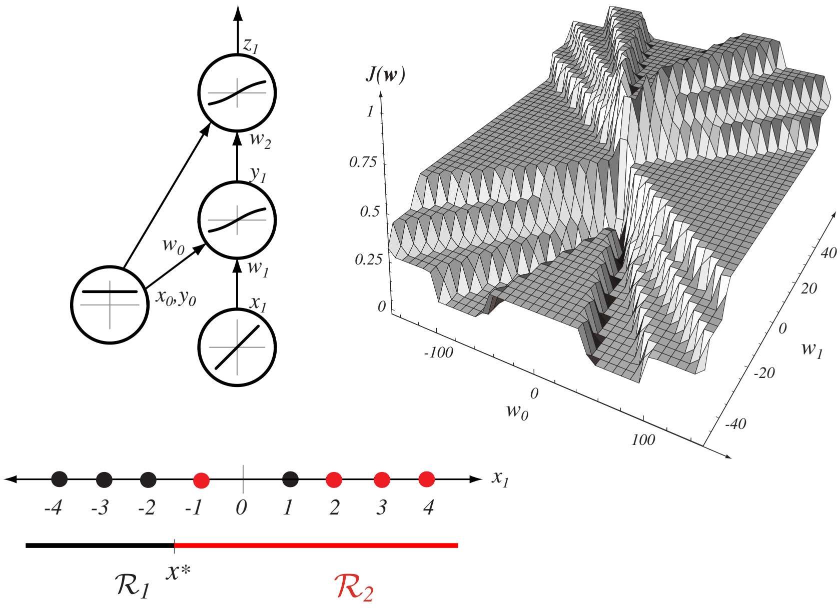

FIGURE 6.8. As in Fig. 6.7, except here the patterns are not linearly separable; the error surface is slightly higher than in that figure. Note too from the error surface that there are two forms of minimum error solution; these correspond to $-2 < x^* < -1$ and $1 < x^* < 2$, in which one pattern is misclassified. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

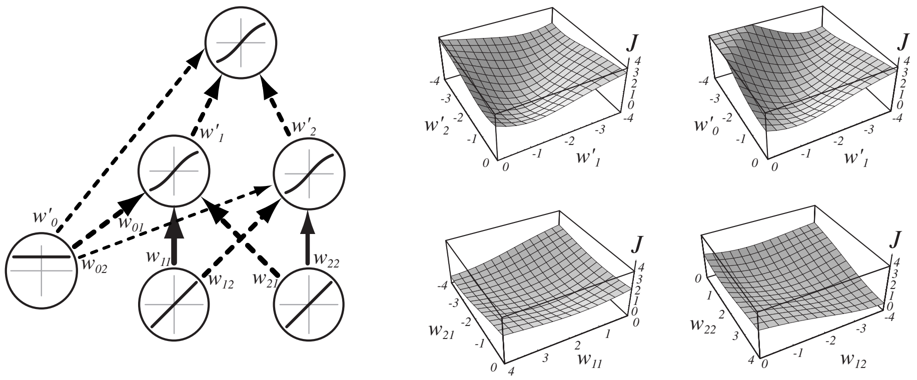

FIGURE 6.9. Two-dimensional slices through the nine-dimensional error surface after extensive training for a 2-2-1 network solving the XOR problem. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

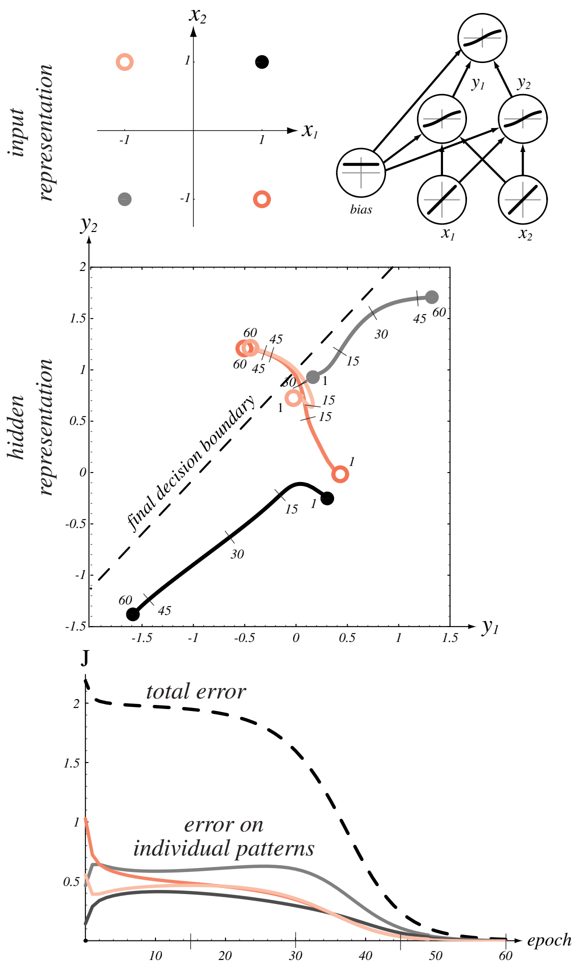

FIGURE 6.10. A 2-2-1 backpropagation network with bias and the four patterns of the XOR problem are shown at the top. The middle figure shows the outputs of the hidden units for each of the four patterns; these outputs move across the $y_1 y_2$-space as the network learns. In this space, early in training (epoch 1) the two categories are not linearly separable. As the input-to-hidden weights learn, as marked by the number of epochs, the categories become linearly separable. The dashed line is the linear decision boundary determined by the hidden-to-output weights at the end of learning; indeed the patterns of the two classes are separated by this boundary. The bottom graph shows the learning curves—the error on individual patterns and the total error as a function of epoch. Note that, as frequently happens, the total training error decreases monotonically, even though this is not the case for the error on each individual pattern. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

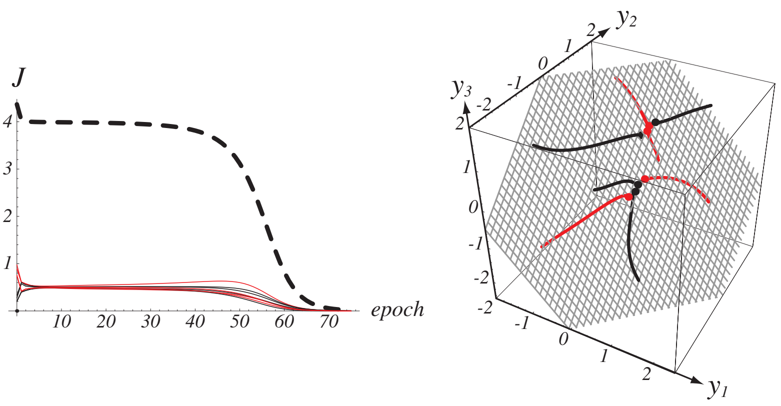

FIGURE 6.11. A 3-3-1 backpropagation network with bias can indeed solve the three-bit parity problem. The representation of the eight patterns at the hidden units ($y_1 y_2 y_3$-space) as the system learns and the planar decision boundary found by the hidden-to-output weights at the end of learning. The patterns of the two classes are indeed separated by this plane, as desired. The learning curve shows the error on individual patterns and the total error J as a function of epoch. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

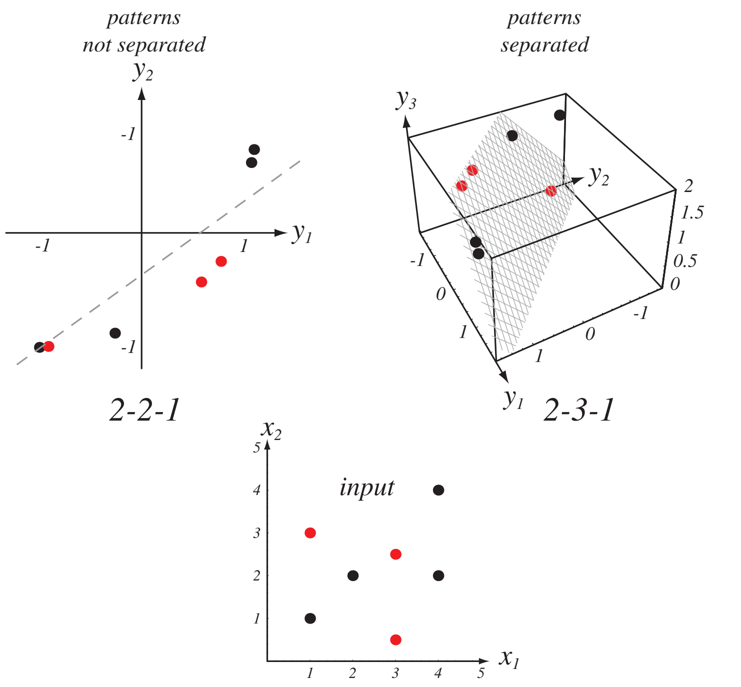

FIGURE 6.12. Seven patterns from a two-dimensional two-category nonlinearly separable classification problem are shown at the bottom. The figure at the top left shows the hidden unit representations of the patterns in a 2-2-1 sigmoidal network with bias fully trained to the global error minimum; the linear boundary implemented by the hidden-to-output weights is marked as a gray dashed line. Note that the categories are almost linearly separable in this $y_1 y_2$-space, but one training point is misclassified. At the top right is the analogous hidden unit representation for a fully trained 2-3-1 network with bias. Because of the higher dimension of the hidden layer representation, the categories are now linearly separable; indeed the learned hidden-to-output weights implement a plane that separates the categories. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

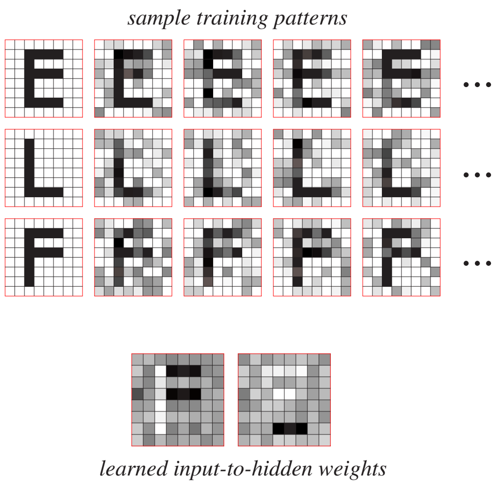

FIGURE 6.13. The top images represent patterns from a large training set used to train a 64-2-3 sigmoidal network for classifying three characters. The bottom figures show the input-to-hidden weights, represented as patterns, at the two hidden units after training. Note that these learned weights indeed describe feature groupings useful for the classification task. In large networks, such patterns of learned weights may be difficult to interpret in this way. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

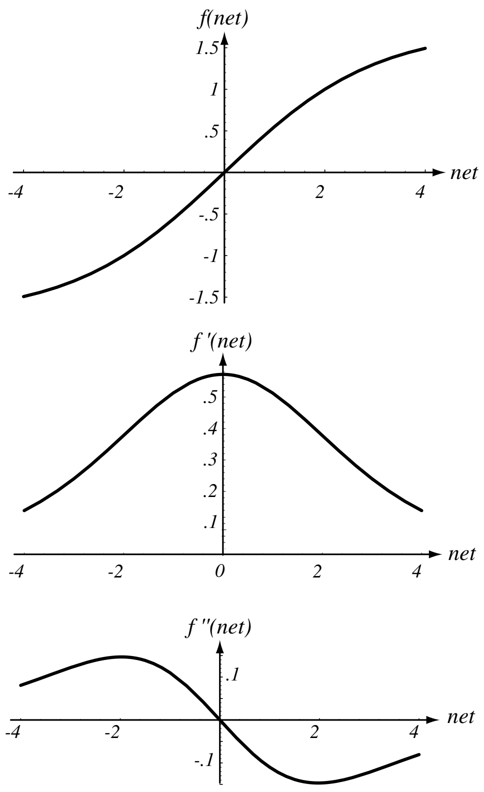

FIGURE 6.14. A useful activation function $f(net)$ is an anti-symmetric sigmoid. For the parameters given in the text, $f(net)$ is nearly linear in the range $-1 < net < +1$ and its second derivative, $f''(net)$, has extrema near $net \simeq \pm 2$. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

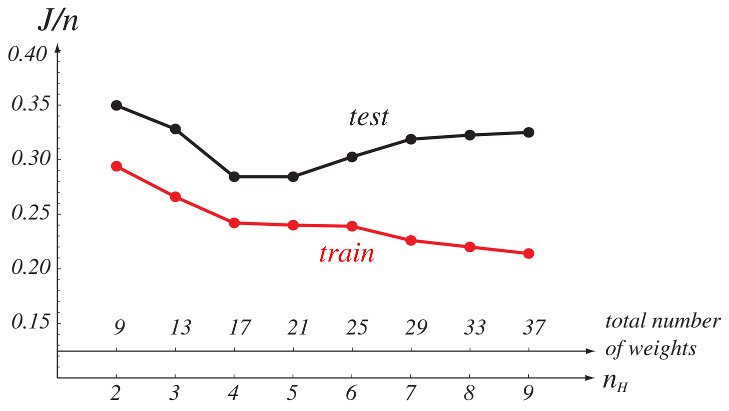

FIGURE 6.15. The error per pattern for networks fully trained but differing in the numbers of hidden units, n_H. Each $2 - n_H - 1$ network with bias was trained with 90 two-dimensional patterns from each of two categories, sampled from a mixture of three Gaussians, and thus $n = 180$. The minimum of the test error occurs for networks in the range $4 \leq n_H \leq 5$, i.e., the range of weights 17 to 21. This illustrates the rule of thumb that choosing networks with roughly $n/10$ weights often gives low test error. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

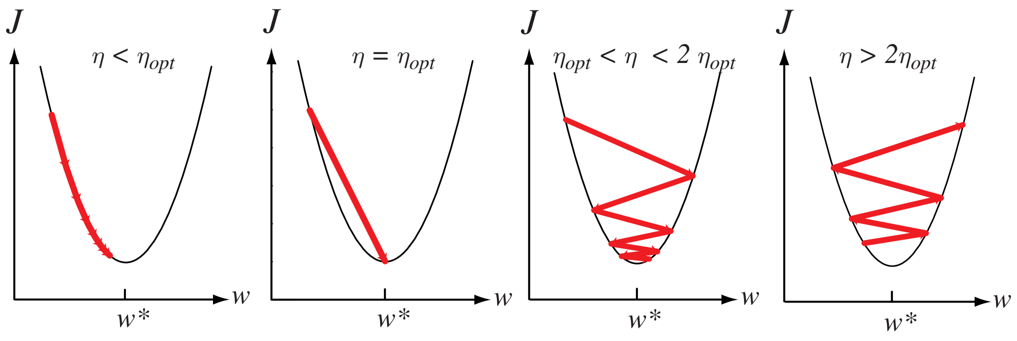

FIGURE 6.16. Gradient descent in a one-dimensional quadratic criterion with different learning rates. If $\eta < \eta_{opt}$, convergence is assured, but training can be needlessly slow. If $\eta = \eta_{opt}$, a single learning step suffices to find the error minimum. If $\eta_{opt} < \eta < 2\eta_{opt}$, the system will oscillate but nevertheless converge, but training is needlessly slow. If $\eta > 2\eta_{opt}$, the system diverges. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

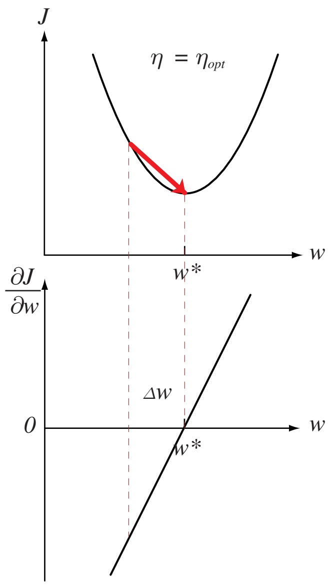

FIGURE 6.17. If the criterion function is quadratic (above), its derivative is linear (below). The optimal learning rate η_{opt} ensures that the weight value yielding minimum error, w^*, is found in a single learning step. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

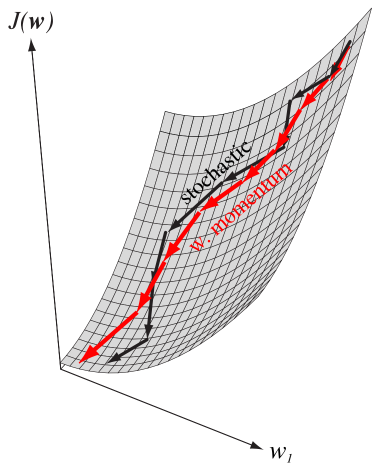

FIGURE 6.18. The incorporation of momentum into stochastic gradient descent by Eq. 37 (red arrows) reduces the variation in overall gradient directions and speeds learning. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

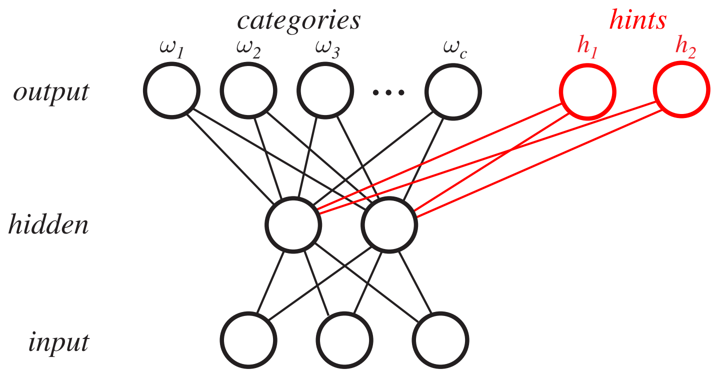

FIGURE 6.19. In learning with hints, the output layer of a standard network having *c* category units is augmented with hint units. During training, the target vectors are also augmented with signals for the hint units. In this way the input-to-hidden weights learn improved feature groupings. The hint units are not used during classification, and thus they and their hidden-to-output weights are removed from the trained network. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

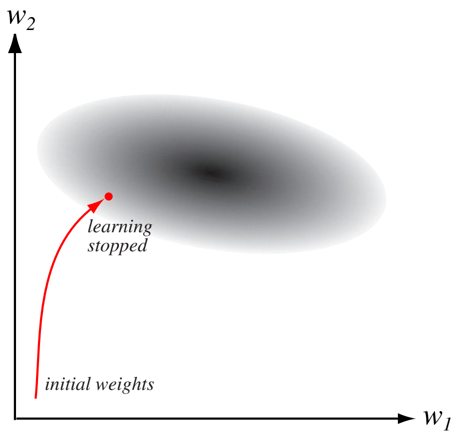

FIGURE 6.20. When weights are initialized with small magnitudes, stopped training leads to final weights that are smaller than they would be after extensive training. As such, stopped training behaves much like a form of weight decay. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

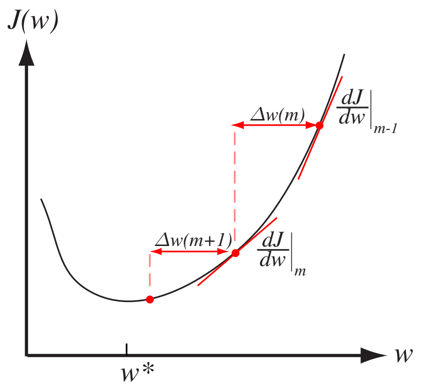
$J(w)$
$\Delta w(m)$
$\left.\frac{dJ}{dw}\right|_{m-1}$
$\Delta w(m+1)$
$\left.\frac{dJ}{dw}\right|_{m}$
w
w^*

FIGURE 6.21. The quickprop weight update takes the error derivatives at two points separated by a known amount, and by Eq. 53 computes the next weight value. If the error can be fully expressed as a second-order function, then the weight update leads to the weight (w^*) leading to minimum error. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

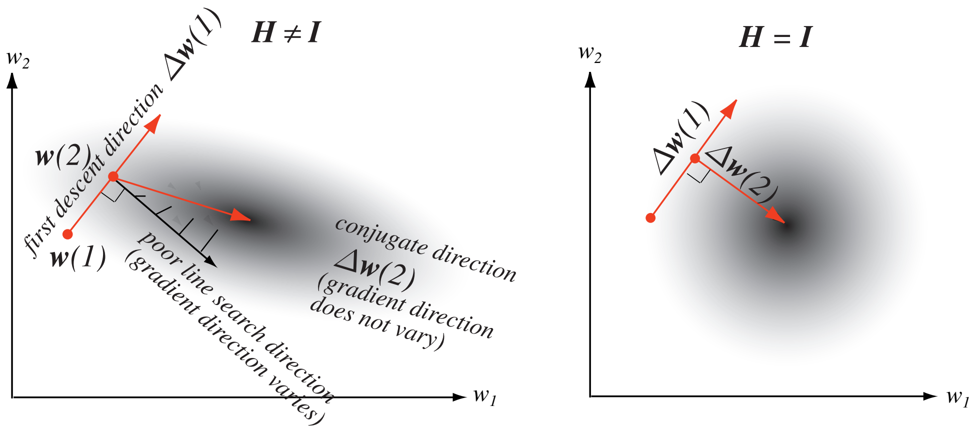

FIGURE 6.22. Conjugate gradient descent in weight space employs a sequence of line searches. If $\Delta\mathbf{w}(1)$ is the first descent direction, the second direction obeys $\Delta\mathbf{w}^t(1)\mathbf{H}\Delta\mathbf{w}(2) = 0$. Note especially that along this second descent, the gradient changes only in magnitude, not direction; as such, the second descent does not "spoil" the contribution due to the previous line search. In the case where the Hessian is diagonal (right), the directions of the line searches are orthogonal. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

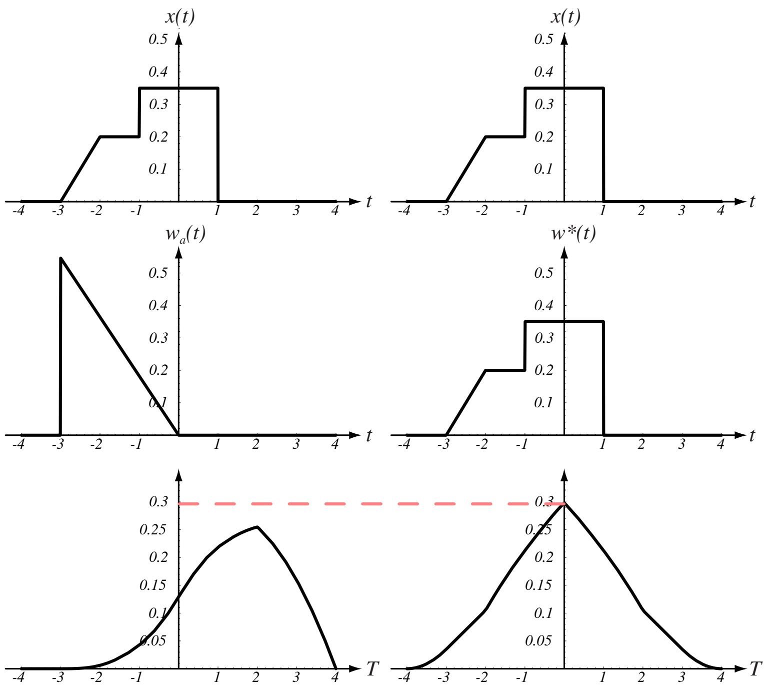

FIGURE 6.23. The left column shows a signal $x(t)$; beneath it is an arbitrary response function $w_a(t)$. At the bottom is the response of the filter as a function of the offset T, as given by Eq. 62. The right column shows the the case where the input and response function "match." The two response functions, $w_a(t)$ and $w^*(t)$ here have the same energy. Note particularly at the bottom that the maximum output in this case is greater than in the nonmatching case at the left. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

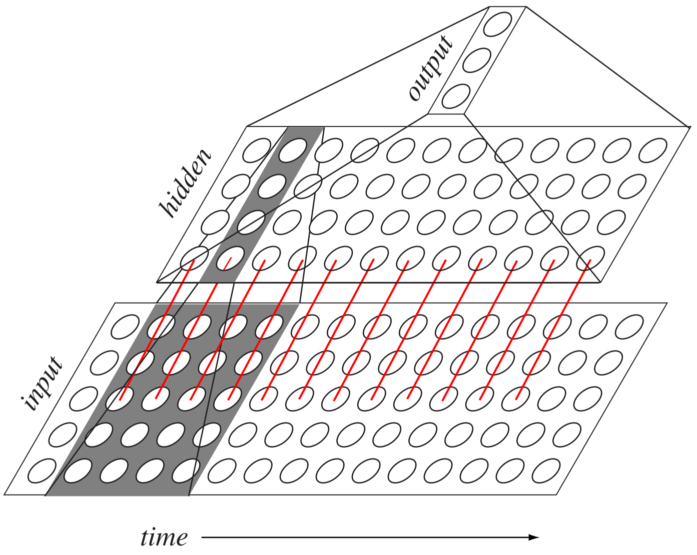

FIGURE 6.24. A time delay neural network (TDNN) uses weight sharing to ensure that patterns are recognized regardless of shift in one dimension; in practice, this dimension generally corresponds to time. Thus all weights shown in red are forced to have the same value. In this example, there are five input units at each time step. Because we hypothesize that the input patterns are of four time steps or less in duration, each of the hidden units at a given time step accepts inputs from only $4 \times 5 = 20$ input units, as highlighted in gray. An analogous translation constraint is also imposed between the hidden and output layer units. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

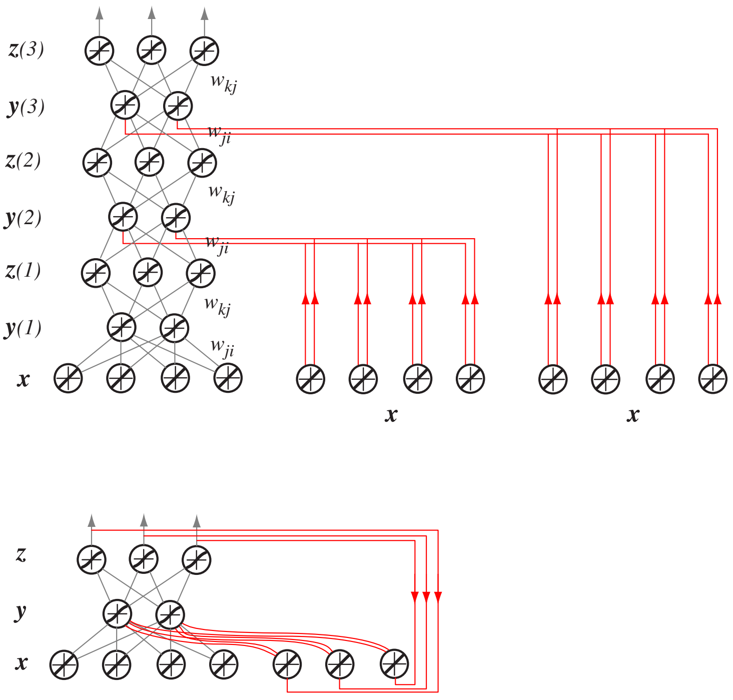

FIGURE 6.25. The form of recurrent network most useful for static classification has the architecture shown at the bottom, with the recurrent connections in red. It is functionally equivalent to a static network with many hidden layers and extensive weight sharing, as shown above. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

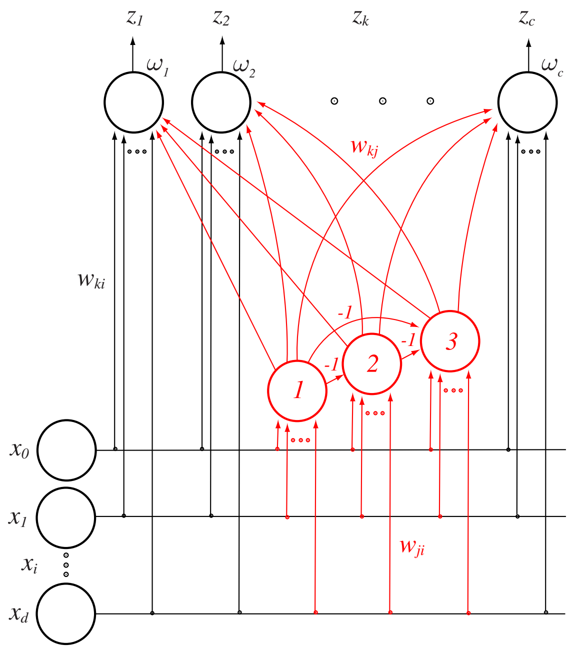

FIGURE 6.26. The training of a multilayer network via cascade-correlation begins with the input later fully connected to the output layer (black). Such weights, w_{ki}, are fully trained using an LMS criterion, as discussed in Chapter 5. If the resulting training error is not sufficiently low, a first hidden unit (labeled 1, in red) is introduced, fully interconnected from the input layer and to the output layer. These new red weights are fully trained, while the previous (black) ones are held fixed. If the resulting training error is still not sufficiently low, a second hidden unit (labeled 2) is likewise introduced, fully interconnected; it also receives a the output from each previous hidden unit, multiplied by -1. Training proceeds in this way, training successive hidden units until the training error is acceptably low. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

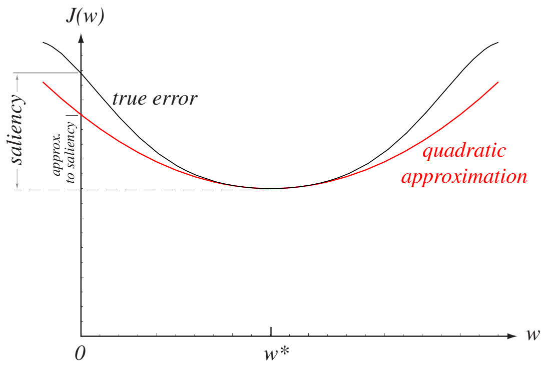

FIGURE 6.27. The saliency of a parameter, such as a weight, is the increase in the training error when that weight is set to zero. One can approximate the saliency by expanding the true error around a local minimum, w^*, and setting the weight to zero. In this example the approximated saliency is smaller than the true saliency; this is typically, but not always, the case. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

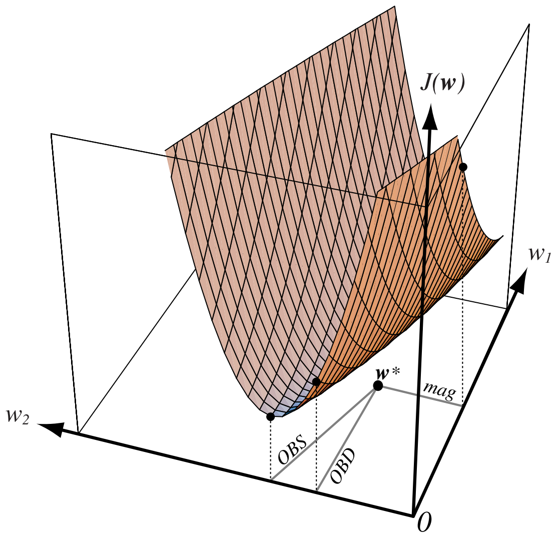

FIGURE 6.28. The figure shows a quadratic error surface as a function of weights, $J(\mathbf{w})$ and the global minimum at $\mathbf{w}^*$. In the second-order approximation to the criterion function, Optimal Brain Damage assumes the Hessian matrix is diagonal, while Optimal Brain Surgeon uses the full Hessian matrix. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.