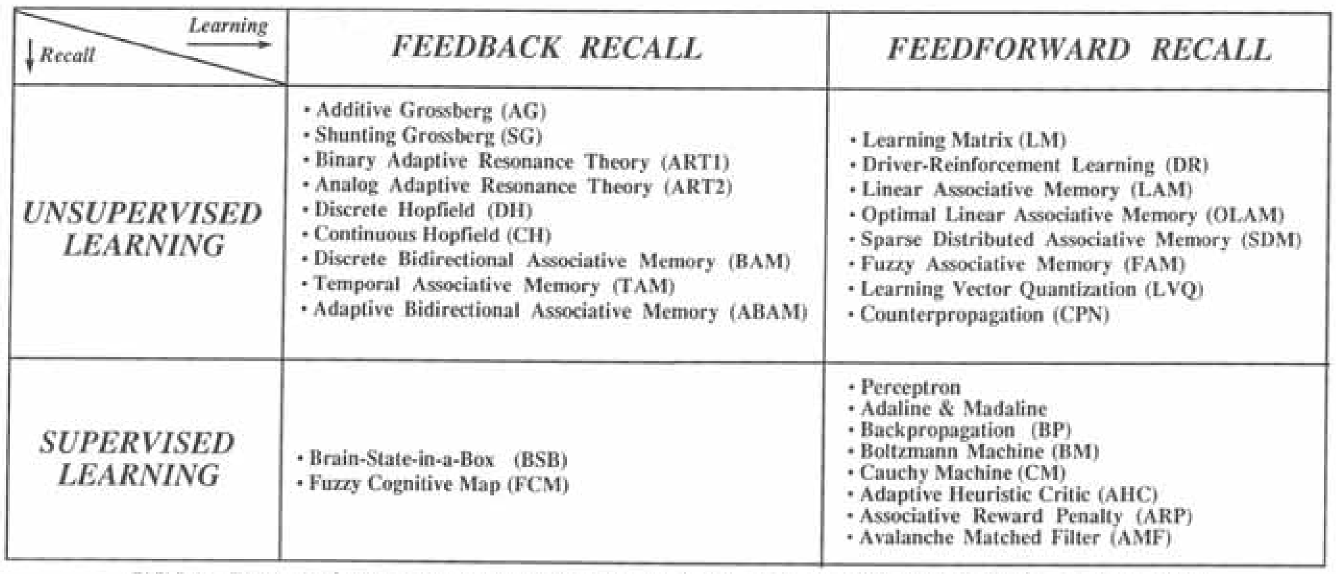
Multilayer Perceptrons (MLPs)

- Appropriate Problems for Neural Network Learning

. Instances are represented by many attribute-value pairs.
. The target function output may be discrete-valued, real-valued, or a vector of several real- or discrete-valued attributes.
. The training examples may contain errors.
. Long training times are acceptable.
. Fast evaluation of learned target function may be required.
. The ability of humans to understand the learned target function is not important.

- Classification of Artificial Neural Networks (ANNs)



Learning → / ↓ *Recall*	*FEEDBACK RECALL*	*FEEDFORWARD RECALL*
UNSUPERVISED LEARNING	• Additive Grossberg (AG) • Shunting Grossberg (SG) • Binary Adaptive Resonance Theory (ART1) • Analog Adaptive Resonance Theory (ART2) • Discrete Hopfield (DH) • Continuous Hopfield (CH) • Discrete Bidirectional Associative Memory (BAM) • Temporal Associative Memory (TAM) • Adaptive Bidirectional Associative Memory (ABAM)	• Learning Matrix (LM) • Driver-Reinforcement Learning (DR) • Linear Associative Memory (LAM) • Optimal Linear Associative Memory (OLAM) • Sparse Distributed Associative Memory (SDM) • Fuzzy Associative Memory (FAM) • Learning Vector Quantization (LVQ) • Counterpropagation (CPN)
SUPERVISED LEARNING	• Brain-State-in-a-Box (BSB) • Fuzzy Cognitive Map (FCM)	• Perceptron • Adaline & Madaline • Backpropagation (BP) • Boltzmann Machine (BM) • Cauchy Machine (CM) • Adaptive Heuristic Critic (AHC) • Associative Reward Penalty (ARP) • Avalanche Matched Filter (AMF)

TABLE 5-1. Taxonomy of ANS paradigms showing the models that belong to each class. See text for a description of the taxonomy.

- Structure of MLPs

. Multiple layers of Perceptrons are connected.
. Three types of layers: input, hidden, and output layers.
Usually, the input layer is not included when we count the number of layers.
. The direction of connection is from the input to output layers (feedforward connection).

. Except the input layer, each unit in the layer is described by

$net_j = \sum_{i=0}^{n} w_{ij} x_i$ and

$o_j = \sigma(net_j) = \frac{1}{1+e^{-net_j}}$

where

w_{ij} = the connection between the ith input to the jth unit,

x_i = the ith input, and

o_j = the output of the jth unit.

Note that

$$\frac{\partial \sigma(net_j)}{\partial net_j} = \sigma_j(net_j)(1-\sigma(net_j)).$$

- Expressive Capabilities of MLPs

. Boolean functions:
Every boolean function can be represented by network with single hidden layer but might require exponential (in number of inputs) hidden units.

. Continuous functions:
Every bounded continuous function can be approximated with arbitrarily small error, by network with one hidden layer (Cybenko 1989; Hornik et al., 1989)

- Backpropagation algorithm

. Error function for multiple outputs:

$$E(\underline{w}) = \frac{1}{2} \sum_{d \in D} \sum_{k \in Outputs} (t_{kd} - o_{kd})^2$$

where D represents the sample index set and $Outputs$ represents the output index set.

. Backpropagation algorithm:
For each training sample d, weights are updated (on-line mode), that is,

$$E_d(\underline{w}) = \frac{1}{2} \sum_{k \in Ouptputs} (t_k - o_k)^2.$$

. two-layer network

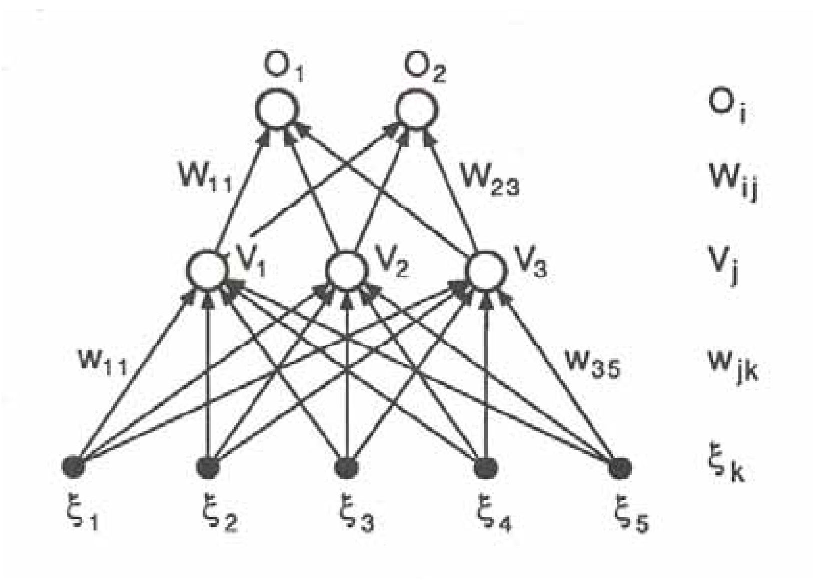


FIGURE 6.1 A two layer feed-forward network, showing the notation for units and weights.

. weight update between hidden and output layers

$v_{kj}^{new} = v_{kj}^{old} + \eta \Delta v_{kj}$ and

$$\Delta v_{kj} = -\frac{\partial E_d}{\partial v_{kj}} = (t_k - o_k) o_k (1 - o_k) h_j = \delta_k h_j$$

where $\delta_k = (t_k - o_k) o_k (1 - o_k)$.

. weight update between input and hidden layers

$w_{ji}^{new} = w_{ji}^{old} + \eta \Delta w_{ji}$ and

$$\Delta w_{ji} = -\frac{\partial E_d}{\partial w_{ji}} = h_j (1 - h_j) \sum_{k \in Outputs} v_{jk} \delta_k x_i = \delta_j^{'} x_i$$

where $\delta_j^{'} = h_j (1 - h_j) \sum_{k \in Outputs} v_{jk} \delta_k$.

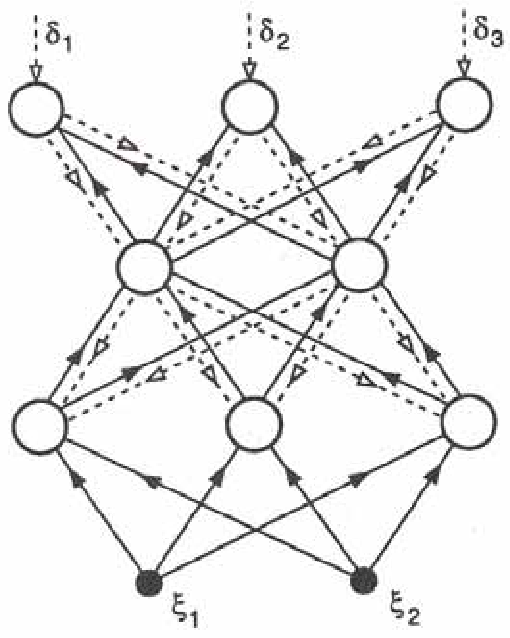


FIGURE 6.2 Back-propagation in a three-layer network. The solid lines show the forward propagation of signals and the dashed lines show the backward propagation of errors (δ's).

- Convergence of Backpropagation

. stochastic gradient descent
. convergence to the global minimum (or even to some local minimum) is not guaranteed. -> usually, small learning rate is applied.
. add momentum term for faster convergence.
. initialize weights near zero -> initial networks near-linear.
. increasingly non-linear functions possible as training progresses

- Backpropgation with momentum term

. A momentum term can be included in the gradient descent algorithm for the fast convergence.

. The weight update term is changed as follows:

$$\Delta w_{ij}(n) = \eta \delta_j x_i + \alpha \Delta w_{ij}(n-1)$$

where α represents the momentum constant.

. Let the time index $t = 0, 1, \cdots, n$. Then,

$$\Delta w_{ij}(n) = \eta \sum_{t=0}^{n} \alpha^{n-t} \delta_j(t) x_i(t) = -\eta \sum_{t=0}^{n} \alpha^{n-t} \frac{\partial E_d}{\partial w_{ij}}.$$

1) for convergence, $|\alpha| < 1$.

2) it tends to accelerate descent in steady downhill directions

since $\Delta w_{ij}(n) \approx -\frac{\eta}{1-\alpha} \frac{\partial E_d}{\partial w_{ij}}$.

3) when $\frac{\partial E_d}{\partial w_{ij}}$ has opposite signs on consecutive iterations,

$\Delta w_{ij}(n)$ shrinks (stabilizing effect).

. Effect of Momentum Term

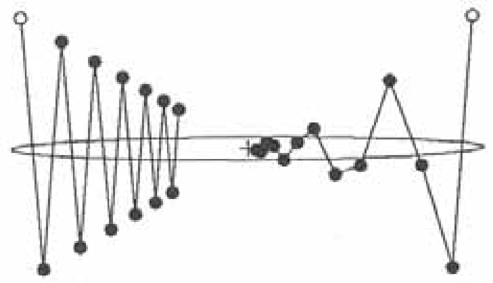


FIGURE 6.3 Gradient descent on the simple quadratic surface of Fig. 5.10. Both trajectories are for 12 steps with $\eta = 0.0476$, the best value in the absence of momentum. On the left there is no momentum ($\alpha = 0$), while $\alpha = 0.5$ on the right.

- Backpropagation with conjugate gradient

. In the CGA, information on Q, that is, the input correlation matrix is required.

. For the on-line mode, an alternative choice of α_k and β_k is needed without the knowledge of Q.

. Determination of α_k:

1) Assuming $E[\alpha_k]$ is unimodal
   (single minimum in the neighborhood of current $\underline{w}_k$),

find three points η_1, η_2, and η_3 such that

$E[\eta_1] \geqq E[\eta_3] \geqq E[\eta_2]$ for $\eta_1 < \eta_2 < \eta_3$.

2) α_k is determined between η_2 and η_3.

. Determination of β_k:

Apply Fletcher-Reeves formula

$$\beta_k = \frac{\underline{g}_{k+1}^T \underline{g}_{k+1}}{\underline{g}_k^T \underline{g}_k}.$$

Conjugate Gradient Algorithm for MLP

Step 1. Set $k=0$, initialize $\underline{w}_0$, and compute $\nabla_k = \frac{\partial E}{\partial \underline{w}}|_{\underline{w}=\underline{w}_k}$.

Step 2. Set $\underline{g}_k = \nabla_k$ and $\underline{d}_k = -\underline{g}_k$.

Step 3. Find α_k that minimizes $E[\underline{w}_k + \alpha_k \underline{d}_k]$.

Step 4. Update the weight parameters: $\underline{w}_{k+1} = \underline{w}_k + \alpha_k \underline{d}_k$.

Step 5. Set $\underline{g}_{k+1} = \nabla_{k+1}$ and $\beta_k = \frac{\underline{g}_{k+1}^T \underline{g}_{k+1}}{\underline{g}_k^T \underline{g}_k}$.

Step 6. Set $\underline{d}_{k+1} = -\underline{g}_{k+1} + \beta_k \underline{d}_k$.

Step 7. If satisfied stop. Otherwise, $k \leftarrow k+1$ and go to Step 3.

. Comparison with the momentum term:

1) weight update rule with momentum term

$$\underline{w}_{k+1} = \underline{w}_k + \triangle \underline{w}_k = \underline{w}_k - \eta \nabla E(\underline{w}_k) + \alpha \triangle \underline{w}_{k-1}$$

2) weight update rule with conjugate gradient:

$$\underline{w}_{k+1} = \underline{w}_k + \triangle \underline{w}_k = \underline{w}_k - \alpha_k \nabla E(\underline{w}_k) + \alpha_k \beta_k \triangle \underline{w}_{k-1}$$

* similar effect with the momentum term

* same computational complexity

* in practice, better solution of parameters

- overfitting in MLPs

. Split samples into training and validation sets

. Let M and N represent the number of parameters and the number of samples respectively. Then, overfitting may occur when $N < 30M$. (Amari, 1996)

. Let r be split ratio between training and validation sets. Then, the optimal split ratio for learning is

$$r_{opt} = 1 - \frac{\sqrt{2M-1}-1}{2(M-1)} \approx 1 - \frac{1}{\sqrt{2M}} \quad \text{(for large } M \text{)}.$$

For example, if $M = 100$, then $r_{opt} = 0.07$, that is, 93% of the samples are alloted to training samples. If $N > 30M$, exhaustive learning is satisfactory.

. overfitting effect on function approximation

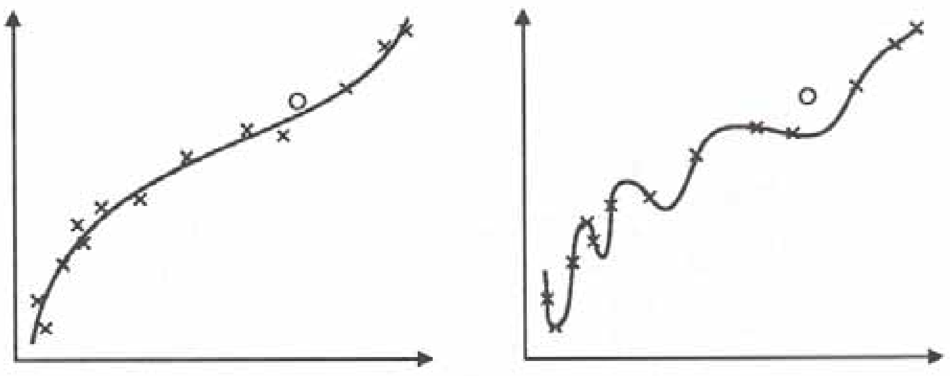


FIGURE 6.13 (a) A good fit to noisy data. (b) Overfitting of the same data: the fit is perfect on the "training set" (x's), but is likely to be poor on a "test set" represented by the circle.

- Complexity regularization

. Add the complexity term to the error measure and minimize the total risk.

. The total risk is described by

$$R(\underline{w}) = E(\underline{w}) + \lambda E_c(\underline{w})$$

where $E(\underline{w})$ represents the error measure,

$E_c(\underline{w})$ represents the complexity term, and

λ represents the regularization parameter.

. weight decay method (Hinton, 1989)

The complexity term is described by

$$E_c(\underline{w}) = \| \underline{w} \|^2.$$

This term is related to the complexity of the hypothesis space.

. weight elimination method (Weigend, 1991)

The complexity term can be bounded as follows:

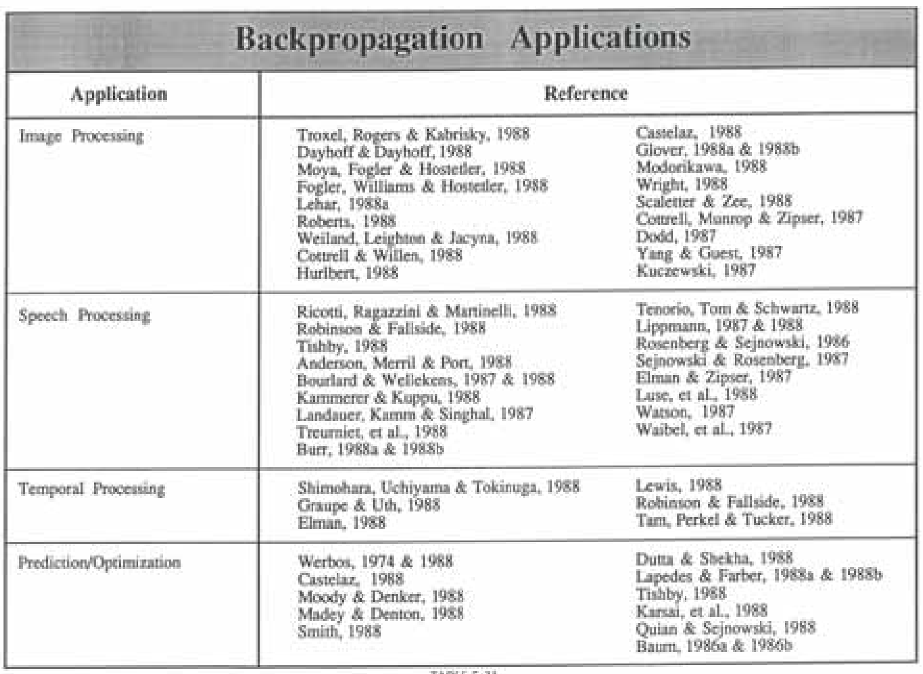
$$E_c(\underline{w}) = \sum_{i \in C} \frac{(w_i/w_0)^2}{1+(w_i/w_0)^2}$$

where C represents all the synaptic connections.

If $|w_i| \ll w_0$, the ith synaptic weight tends to be eliminated.

If $|w_i| \gg w_0$, the ith synaptic weight has major influence to the network.

- Applications of MLPs



Backpropagation Applications

Application	Reference	
Image Processing	Troxel, Rogers & Kabrisky, 1988 Dayhoff & Dayhoff, 1988 Moya, Fogler & Hostetler, 1988 Fogler, Williams & Hostetler, 1988 Lehar, 1988a Roberts, 1988 Weiland, Leighton & Jacyna, 1988 Cottrell & Willen, 1988 Hurlbert, 1988	Castelaz, 1988 Glover, 1988a & 1988b Modorikawa, 1988 Wright, 1988 Scaletter & Zee, 1988 Cottrell, Munrop & Zipser, 1987 Dodd, 1987 Yang & Guest, 1987 Kuczewski, 1987
Speech Processing	Ricotti, Ragazzini & Martinelli, 1988 Robinson & Fallside, 1988 Tishby, 1988 Anderson, Merril & Port, 1988 Bourlard & Wellekens, 1987 & 1988 Kammerer & Kuppu, 1988 Landauer, Kamm & Singhal, 1987 Treurniet, et al., 1988 Burr, 1988a & 1988b	Tenorio, Tom & Schwartz, 1988 Lippmann, 1987 & 1988 Rosenberg & Sejnowski, 1986 Sejnowski & Rosenberg, 1987 Elman & Zipser, 1987 Luse, et al., 1988 Watson, 1987 Waibel, et al., 1987
Temporal Processing	Shimohara, Uchiyama & Tokinuga, 1988 Graupe & Uth, 1988 Elman, 1988	Lewis, 1988 Robinson & Fallside, 1988 Tam, Perkel & Tucker, 1988
Prediction/Optimization	Werbos, 1974 & 1988 Castelaz, 1988 Moody & Denker, 1988 Madey & Denton, 1988 Smith, 1988	Dutta & Shekha, 1988 Lapedes & Farber, 1988a & 1988b Tishby, 1988 Karsai, et al., 1988 Quian & Sejnowski, 1988 Baum, 1986a & 1986b

TABLE 5-21.

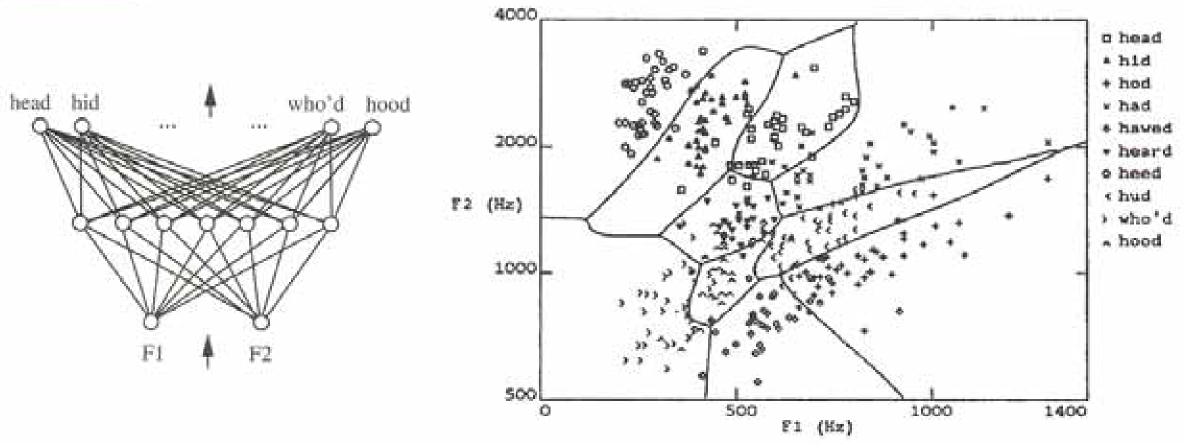


FIGURE 4.5
Decision regions of a multilayer feedforward network. The network shown here was trained to recognize 1 of 10 vowel sounds occurring in the context "h_d" (e.g., "had," "hid"). The network input consists of two parameters, F1 and F2, obtained from a spectral analysis of the sound. The 10 network outputs correspond to the 10 possible vowel sounds. The network prediction is the output whose value is highest. The plot on the right illustrates the highly nonlinear decision surface represented by the learned network. Points shown on the plot are test examples distinct from the examples used to train the network. (Reprinted by permission from Haung and Lippmann (1988).)

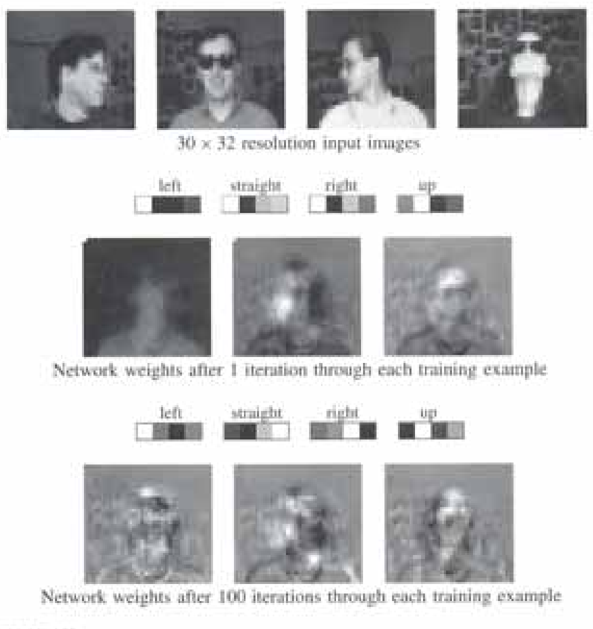

30 × 32 resolution input images
left
straight
right
up
Network weights after 1 iteration through each training example
left
straight
right
up
Network weights after 100 iterations through each training example

FIGURE 4.10
Learning an artificial neural network to recognize face pose. Here a 960 × 3 × 4 network is trained on grey-level images of faces (see top), to predict whether a person is looking to their left, right, ahead, or up. After training on 260 such images, the network achieves an accuracy of 90% over a separate test set. The learned network weights are shown after one weight-tuning iteration through the training examples and after 100 iterations. Each output unit (left, straight, right, up) has four weights, shown by dark (negative) and light (positive) blocks. The leftmost block corresponds to the weight w_0, which determines the unit threshold, and the three blocks to the right correspond to weights on inputs from the three hidden units. The weights from the image pixels into each hidden unit are also shown, with each weight plotted in the position of the corresponding image pixel.