Population coding, where neurons with broad and overlapping firing rate tuning curves collectively encode information about a stimulus, is a common feature of sensory systems. We use decoding methods and measured properties of NMDA-mediated LTP induction to study the impact of long-term potentiation of synapses between the neurons of such a coding array. We find that, due to a temporal asymmetry in the induction of NMDA-mediated LTP, firing patterns in a neuronal array that initially represent the current value of a sensory input will, after training, provide an experienced-based prediction of that input instead. We compute how this prediction arises from and depends on the training experience. We also show how the encoded prediction can be used to generate learned motor sequences, such as the movement of a limb. This involves a novel form of memory recall that is driven by the motor response so that it automatically generates new information at a rate appropriate for the task being performed.

Although it is widely believed that long-term potentiation (LTP) plays a central role in learning and memory, the specific mechanisms by which changes in synaptic efficacy affect behavior are unknown. Synaptic modification rules inferred from experiments on LTP suggest that the pattern of synaptic potentiation produced by training experience reflects correlations in the firing patterns of neurons. How is the information stored in these synaptic enhancements read out to affect future behavior? A much-studied conjecture is that the stored information is encoded in the firing patterns of fixed-point attractors of the neural network dynamics (Marr, 1971; Hopfield, 1982; Kohonen, 1984, 1988; Grossberg, 1988; Amit, 1989; Rolls, 1989; Hertz et al., 1991). In these models memory recall is associative and quasi-static. We provide another suggestion, that long-term potentiation allows a neural network to generate continuously from sensory input data an experienced-based prediction of the future value of an encoded quantity (Levy, 1989; Muller and Kubie, 1989). Our proposal relies on a specific prediction that can be tested experimentally.

We model how the value of an ensemble-coded quantity is affected by potentiation of recurrent collateral synapses between neurons within the coding array. Our computational procedure is to decode place cell activity before and after a simulated training period. The synaptic changes that occur during the training period are computed using characteristics of NMDA-mediated LTP observed in hippocampal slice preparations (Levy and Steward, 1983; Gustafsson et al., 1987; Bliss and Collingridge, 1993; Malenka and Nicoll, 1993). We then calculate how these training-induced changes of synaptic strength alter network firing, and decode the modified firing patterns. We find that training experience produces a shift in the information encoded by ensemble firing activity that has clear behavioral implications. Our calculations provide specific predictions that can be tested experimentally.

Since our results are based on general properties of LTP and on a common form of population coding, they are applicable to a wide variety of neural systems. Examples to which our ideas could be applied include neurons responding to the position or velocity of a limb (Kalaska et al., 1983), neurons characterizing the motion of a visual image (Mausell and Newsome, 1987; Steinmetz et al., 1987) or location of a sound source (Knudsen and Konishi, 1978; Konishi, 1991), or hippocampal place cells representing spatial position within the environment (O'Keefe and Dostovsky, 1971; O'Keefe and Nedel, 1978; O'Keefe, 1979). In many cases it has proven possible to decode the information represented by neural firing in such arrays (for reviews see Abbott, 1994; Salinas and Abbott, 1994) so that an estimate of the encoded quantity can be computed from measured firing rates. Georgopoulos and collaborators developed one such approach by defining the population vector relating activity in the motor cortex of the monkey to the direction of arm movements (Georgopoulos et al., 1986, 1988). Wilson and McNaughton (1993) have decoded the output of place cells in the hippocampus of the rat to determine the spatial location represented by their firing.
Decoding the particular set of firing rates \( r_i \), is the "center of gravity" of the neuronal vector (Caminiti et al., 1991; Young and Yamane, 1992; Salinas and Abbott, 1994). This is the least-squares method where \( \hat{\beta} \) is defined as the vector giving the best fit of the average firing rates \( f_i \) to the observed rates \( r_i \), so that

\[
\sum (r_i - f_i(\hat{\beta}))^2 = \text{minimum}.
\]

Both methods give similar, though not identical, results.

Synaptic Modification Rule

Our basic program is to determine how synaptic modifications induced by training experience affect the population vector that is ensemble coded by network firing. To do this we need to model how training modifies synapses. As mentioned previously, we are concerned with experience-induced changes in synaptic weights rather than with the values of the weights themselves. An advantage of this approach is that we can consider small shifts in synaptic weights as a perturbation on the basic neuronal response characterized by firing rate tuning curves (Abbott, 1994). This allows us to compute the effects of training-induced LTP without making large numbers of assumptions. Our calculations are quantitatively accurate only when the amount of LTP is fairly small, although we expect our results to be qualitatively correct even for larger changes in synaptic weights. Because the mechanisms we consider are collective, even small changes in synaptic weights can produce behaviorally relevant effects. We associate the synaptic weights we are discussing with individual synapses but a more abstract interpretation in terms of "effective" synaptic strengths is also possible. Suppose that during the training period the sensory input takes a specific sequence of values given by \( X(t) \). This could represent the presentation of a temporal sequence of sensory stimuli with specific attributes, for example, the movement of a specific trajectory for proprioceptive coding. The training sequence may include repetitions. If the average firing rate tuning curves \( f_i \) have been measured we can predict how each neuron will fire (on average) during the training period,
amount of synaptic potentiation is small and then the constraint does not play an important role (see also MacKay and Miller, 1990).

Results

**Experience-Based Prediction**

We will assume that, before training, the activity of the neuronal array accurately represents the value of the sensory input. Thus, initially the sensory input vector and the population vector are identical, \( \bar{p} = \bar{x} \). Indeed, if the tuning curves of the encoding neurons are evenly distributed over the range of coded values, it can be shown that the population vector given by Equation 1 will agree with the value of the sensory input (Georgopoulos et al., 1988). However, modifications in the strength of synapses between coding neurons produced by training experience can change the population vector characterizing the ensemble response to an input and destroy this equality. At first sight, this might appear to introduce unwanted inaccuracy in the representation of the sensory input, but we will see that the shifts in ensemble-coded quantities caused by experience-based synaptic modification can be useful.

How do the synaptic weight changes arising from training alter the pattern of neuronal firing produced by exposure to an arbitrary sensory input after training has concluded? We assume that potentiation of a synapse increases the firing rate of the postsynaptic neuron by an amount given by the product of the synaptic weight change times the firing rate of the corresponding presynaptic neuron. The total effect for a given postsynaptic neuron is obtained by summing over all of its presynaptic neurons. Thus, after training, the average firing rate of neuron \( i \) when the input takes the value \( \bar{x} \) is, to linear order in \( \Delta W \),

\[
    r_i = f_c(\bar{x}) + \sum_j \Delta W_{ij} f_j(\bar{x})
\]

rather than \( r_i = f_c(\bar{x}) \) as it was before training. We now compute how these training-induced changes in the firing rates shift the value of the population vector representing the ensemble firing. We do this by inserting the rates given by Equation 5 into Equation 1. To linear order in \( \Delta W \) we find that, after training, when the sensory input takes the value \( \bar{x} \) the responses of the network neurons code for a population vector

\[
    \bar{p} = \bar{x} + \frac{\sum_j (\Delta x_{ij} \Delta W_{ij} f_j(\bar{x}))}{\sum_j f_j(\bar{x})}
\]

that is different from the sensory input vector \( \bar{x} \). This result is derived in Appendix A. A similar result using the least-squares method of decoding is also given in Appendix A.

What is the significance of the fact that, after training, the coded population vector is no longer equal to the sensory input? Before presenting the results of detailed calculations we will answer this question qualitatively. During training, neurons responding to the training inputs are sequentially activated. Because of the temporal asymmetry of NMDA-mediated LTP induction, synapses from presynaptic cells activated at one point in the training sequence to postsynaptic cells activated simultaneously or somewhat later will be potentiated. Now consider what happens after the training period is over. Suppose that one of the sensory inputs from the training sequence is presented to the network. The population vector is also shifted for sensory inputs not in the training set. Indeed, if the tuning curves of the encoding neurons are evenly distributed over the range of coded values, it can be shown that the population vector given by Equation 1 will agree with the value of the sensory input (Georgopoulos et al., 1988). However, modifications in the strength of synapses between coding neurons produced by training experience can change the population vector characterizing the ensemble response to an input and destroy this equality. At first sight, this might appear to introduce unwanted inaccuracy in the representation of the sensory input, but we will see that the shifts in ensemble-coded quantities caused by experience-based synaptic modification can be useful.

How do the synaptic weight changes arising from training alter the pattern of neuronal firing produced by exposure to an arbitrary sensory input after training has concluded? We assume that potentiation of a synapse increases the firing rate of the postsynaptic neuron by an amount given by the product of the synaptic weight change times the firing rate of the corresponding presynaptic neuron. The total effect for a given postsynaptic neuron is obtained by summing over all of its presynaptic neurons. Thus, after training, the average firing rate of neuron \( i \) when the input takes the value \( \bar{x} \) is, to linear order in \( \Delta W \),

\[
    r_i = f_c(\bar{x}) + \sum_j \Delta W_{ij} f_j(\bar{x})
\]

rather than \( r_i = f_c(\bar{x}) \) as it was before training. We now compute how these training-induced changes in the firing rates shift the value of the population vector representing the ensemble firing. We do this by inserting the rates given by Equation 5 into Equation 1. To linear order in \( \Delta W \) we find that, after training, when the sensory input takes the value \( \bar{x} \) the responses of the network neurons code for a population vector

\[
    \bar{p} = \bar{x} + \frac{\sum_j (\Delta x_{ij} \Delta W_{ij} f_j(\bar{x}))}{\sum_j f_j(\bar{x})}
\]

that is different from the sensory input vector \( \bar{x} \). This result is derived in Appendix A. A similar result using the least-squares method of decoding is also given in Appendix A.

What is the significance of the fact that, after training, the coded population vector is no longer equal to the sensory input? Before presenting the results of detailed calculations we will answer this question qualitatively. During training, neurons responding to the training inputs are sequentially activated. Because of the temporal asymmetry of NMDA-mediated LTP induction, synapses from presynaptic cells activated at one point in the training sequence to postsynaptic cells activated simultaneously or somewhat later will be potentiated. Now consider what happens after the training period is over. Suppose that one of the sensory inputs from the training sequence is presented to the network. The population vector will not be equal to the sensory input vector but rather to a value somewhat ahead of it in the training sequence. Thus, the population vector representing ensemble activity no longer encodes the present value of the sensory input, but instead, provides a prediction of its future value based on the training experience.

The population vector is also shifted for sensory inputs that are not in the training sequence but are similar to training inputs. As we will see from the more detailed computations, this shift is toward the training set (see also Muller et al., 1991). As a result, the network can interpolate to provide predictions for sensory inputs not in the training set.

The ability of LTP to produce an ensemble-coded prediction on the basis of training experience is seen in Figure 1. We consider a two-dimensional sensory input vector so that our results can be plotted and visualized easily. This vector could correspond, for example, to the position or movement direction of a limb. In Figure 1, we show an area in the two-dimensional space of sensory inputs. Initially, the response of the coding array to sensory inputs anywhere within this area resulted in a population vector that was equal to the sensory input vector, \( \bar{p} = \bar{x} \). We then simulated a particular training sequence. In these plots, a training sequence is represented by a path through the two-dimensional space of sensory inputs. In the case of arm position coding, these paths would represent arm movement trajectories. The solid lines in Figure 1 indicate the particular training sequences used. These training sequences induce changes in synaptic efficacy as discussed in the Materials and Methods section.

After training, when the network was exposed to a sensory input vector \( \bar{x} \) and its response population vector was computed, the result was no longer \( \bar{p} = \bar{x} \), but rather, for most \( \bar{x} \) values within the area shown in the figures, the population vector shifted relative to \( \bar{x} \). In Figure 1, the arrows indicate these shifts, that is, the arrows are equal to the difference \( \bar{p} - \bar{x} \). This difference is defined for every value of \( \bar{x} \), but we have plotted the arrows on a grid of \( \bar{x} \) values. At every point on the grid, the tail of the arrow represents a particular sensory input to the network, and the head of the arrow marks the corresponding value of the population vector decoded from the network activity evoked by this input. Examination of the arrows indicates that for sensory inputs lying on the training sequence path, the population vector lies ahead of the sensory input vector, forward along the path. Beside the training path, the population vector is displaced both forward and toward the training sequence relative to the actual sensory input.

Figure 1 implies that a network that initially encoded a sensory input will no longer code for its value after training, but instead will provide a prediction of a future value on the basis of the training experience. This is seen in Figure 2. A network was trained using the circular sequence of Figure 1b. Then, the same sequence was used as input to the network and the resulting population vector was decoded. In Figure 2, the dashed line shows the horizontal component of the sensory input vector as a function of time. The solid line indicates the corresponding component of the population vector. The output of the network, as decoded by the population vector, clearly provides a prediction of the actual sequence. Figure 2b shows a prediction that is more advanced in time, though less accurate, than the one in Figure 2a. The network used in Figure 2b had more highly overlapping firing tuning curves than those used in Figure 2a, and this resulted in a longer prediction time (see below).

The arrows in Figure 1 collectively provide a map of the space of sensory input values that predicts how sequences of inputs are likely to flow on the basis of the training sequence. It is possible to build up such maps from multiple
Figure 1. Training-induced shifts in the encoded value of a sensory input. The sensory input vector is represented as a point in a two-dimension plot of size 10 by 10. Initially the ensemble-coded population vector and the sensory input vector are identical (not shown). The solid curves show the training sequences used to induce LTP between encoding neurons. The arrows show the shifts in the encoded population vector arising from this simulated training experience. Arrows are drawn for a grid of different sensory input values. a, The training sequence moved from lower left to upper right at uniform speed along the solid curve. b, The training sequence is represented by a counterclockwise circle with constant speed. c, Two superimposed training sequences are shown. One is the same as a, while the other is a straight line with constant velocity moving from upper left to lower right.

Figure 2. Comparison of actual and coded values after training. The value plotted is the horizontal component of the position in the plane of Figure 1. The network was first trained using the circular sequence of Figure 1b. The same training sequence was then used as a sensory input to the trained network and the resulting population vector was computed. The actual sensory input to the network is plotted as a dashed curve and the corresponding encoded value is indicated by a solid curve. In both figures the coded value leads the actual value. The difference between the upper and lower plots corresponds to a fivefold increase in the overlap of the neuronal tuning curves.

paths. Figure 1c shows the case of two superimposed paths. Clearly, information about both paths is present in the map, although some confusion can arise at the junction. If many training sequence paths are superimposed, information about individual sequences gets lost but, nevertheless, the map contains information about how the sequences collectively flowed through the space of sensory input values. In our study of hippocampal place cell coding we showed how a large number of training paths build up a map that can be used for navigation (Blum and Abbott, 1996). Here we will concentrate primarily on single paths.

How large can the training-induced shift in the population vector be? This depends on the rate of LTP induction and degree of its temporal asymmetry, and on the width and degree of overlap of the neuronal tuning curves (see Appendix B). Because the shift along a learned path is approximately tangent to the path, the shifted population vector acts as a
Figure 3. Movements directed by the training-induced shifts. After training, the networks shown in Figure 1 were used to guide movement. The dashed curves indicate the training sequences (identical to those shown in Fig. 1), while the solid curves show the linear predictor of the future sensory input vector. Broader, more overlapping tuning curves produce large shifts and, consequently, longer prediction times. The resulting linear predictions can be significantly longer than the 100–200 msec time window for LTP. This somewhat surprising result is derived in Appendix B. The magnitude of the shift in the coded population vector is proportional to the product of three factors. One factor is the amount of synaptic potentiation that occurs in one LTP window time between two neurons firing at their maximum rates. The second factor is the average width of the firing rate tuning curves, and the third is a measure of the amount of overlap between the firing rate tuning curves of the coding neurons. If we want to know how far into the future a neuronal array predicts, we need to divide the shift in the population vector by the speed at which sensory inputs are changing. For slowly evolving input data and broad, highly overlapping neuronal tuning curves, large prediction times are possible. Note, however, that the prediction we are discussing is a simple linear extrapolation so long prediction times may produce inaccurate results if the training sequence is too nonlinear over the range of extrapolation.

Calculations (see Appendix B) also show that the size of the shift in the population vector is relatively insensitive to the speed at which the training sequence is presented. This is because rapid sequences result in larger asymmetries even though their shorter periods of firing induce less LTP. As a result, a sequence can be learned at any reasonable presentation speed. The resulting synaptic weight changes store information primarily about the direction and not the magnitude of the “velocity” of the training sequence path.

**Learning Motor Sequences**

The arrows in Figure 1 suggest that the information stored in potentiated synapses and read out by the population vector could be used not only to predict a training sequence but to recreate it as well. To make these ideas concrete, consider coding of arm position (Kalaska et al., 1983). In this case, the plots in Figure 1 represents the position of the arm (hand, for example) during a two-dimensional movement. The training sequence in Figure 1b corresponds to a circular arm movement. The arrows in Figure 1b indicate that from any position the population vector indicates where the arm should go next to generate the learned movement. To test whether this prediction can be used to produce a learned sequence, we simulated a movement guided by the population vector. We started with an arbitrary position and generated a movement path by traveling at constant speed from a given position toward the position coded by the population vector evoked by that value of \( \mathbf{x} \). The solid lines in Figure 3 show the resulting movement path. Figure 3a and 3b shows that the learned movement is quite accurately recreated by this simple procedure. Figure 3c shows that when two paths are superimposed confusion can arise where they cross, but otherwise, the path is followed accurately.

Figure 3 suggests a mechanism by which neuronal activity can guide the execution of a learned motor task. If temporally asymmetric NMDA-mediated LTP occurs between neurons
movement to act as the training sequence for subsequent paths. Figure 4 shows an interesting consequence of this during iterated learning of a movement. When we allow a guided tendency for guided paths to "cut corners" relative to the training motor sequence is stable. If some perturbation knocks the arm off the learned trajectory, the target position that is generated target position. In this scheme, the recall of information about the learned sequence is driven by the motor response, not by internal feedback within the network. The retrieval rate automatically matches the rate at which the task is being performed because it is the movement itself that triggers the recall. As a result, once the task is learned, it can be performed at any desired rate or in discrete partial sequences.

The ability of this mechanism to reproduce a learned sequence is based on the fact the NMDA-mediated LTP automatically produces a predictive representation of the training sequence. It is equally important that the arrows in Figure 1 show a shift toward the training path for sensory inputs beside the training sequence. This indicates that the generated motor sequence is stable. If some perturbation knocks the arm off the learned trajectory, the target position that is generated by the perturbed arm position will direct the motor system back toward the learned path.

Careful examination of Figure 3 reveals that there is a tendency for guided paths to "cut corners" relative to the training paths. Figure 4 shows an interesting consequence of this during iterated learning of a movement. When we allow a guided movement to act as the training sequence for subsequent movements, there is a gradual straightening of the path of the movement. If the process is iterated enough times, straight-line motion will result. Straightening will not occur if LTP is deactivated or if some other constraint prevents the straight-line movement from being carried out. However, this process could give rise to more efficient movement trajectories consistent with physical constraints.

Discussion

LTP appears to have the correlation-dependent properties (Morris et al., 1986; Baudry and Davis, 1991; Hawkins et al., 1993) needed to cause information about a training experience to be stored in synaptic weights (Sejnowski, 1977; Byrne and Berry, 1989; Gluck and Rumelhart, 1990; Levy et al., 1990). However, the mechanisms by which this information is read out to affect behavior have not been clearly established. We suggest that the shifts in ensemble-coded quantities that we have computed provide one such mechanism. Through these shifts, information stored in synaptic weights appears as modifications in firing rates that can be transferred to other neural networks (Salinas and Abbott, 1995). We have shown that the temporal properties of NMDA-mediated LTP naturally cause a population of neurons that represent a quantity to start predicting that quantity after training. Techniques can reveal this phenomenon at the network level, and its molecular basis can be traced to properties of the NMDA receptor (Hestrin et al., 1990; Jahr and Stevens, 1990; Lester et al., 1990; Madison et al., 1991). We find that training experience produces a shift in the position coded by neuronal ensemble activity that has clear predictive implications. The difference between the coded population vector and sensory input vector provides an experience-dependent map of sensory space that automatically develops during training and that can be used to generate experienced-based predictions and extrapolations. Furthermore, this information can be used to generate learned motor sequences. The figures we have constructed provide a graphic image of the information stored in training-modified synaptic weights.

Direct readout from the coded population vector after training on a specific sequence provides the location of the next point in that sequence. However, for some tasks it may be more advantageous to obtain the direction and distance from the present point to the next point in the sequence analogous to the arrows drawn in Figures 1 and 3. To extract this information, a neural circuit must effectively subtract the input vector \( \hat{x} \) from the population vector \( \hat{p} \). Information about both of these quantities is available: \( \hat{x} \) is represented by the input to the coding array, and \( \hat{p} \) is coded by its output. One method for extracting the difference \( \hat{p} - \hat{x} \) (Blum and Abbott, 1996) is based on a mechanism used to determine a gaze-invariant measure of the position of an object in parietal cortex (Andersen et al., 1988; S. Zipser and Andersen, 1988). If information about both \( \hat{p} \) and \( \hat{x} \) is conveyed to and simultaneously represented by a single network, it has been shown that downstream networks can extract the difference \( \hat{p} - \hat{x} \) (Salinas and Abbott, 1995).

Figures 1c and 3c show that multiple sequences can be stored and recalled, but they tend to interfere with each other. Additional information can be provided to resolve ambiguities at junction points, but, clearly, large numbers of sequences cannot be superimposed without loss of information about individual sequences. In our study of hippocampal place-cell coding (Blum and Abbott, 1996) we found that when large numbers of paths were stored the result was a navigational map of the environment, not a record of specific paths. How, then, are large numbers of sequences stored? One possibility is the random reassignment of turning curves as occurs for place cells when a rat enters a new environment. If a training sequence is stored and recalled as we have suggested, the...
reassignment of coding cells would effectively remove the shifts in the population vector that represent the learned sequence. This is because the coherent summation of synaptic weights that produces these shifts is destroyed by the reassignment. A new sequence can then be learned using the new assignment of coding cells. However, since the original pattern of synaptic weight changes has not been erased, this is a nondestructive process. Restoration of the original tuning curves of the coding neurons causes the first sequence to reappear. Through this mechanism a large number of different sequences can be stored simultaneously by shifting and shuffling the tuning curves of the coding neurons.

In our model, the shifts that store a particular sequence arise because network output is decoded using the pretraining firing rate tuning curves \( f \) even after the firing rates have been modified by training. Both before and after training, \( f \) is used to define \( \tilde{s} \) in Equation 1 and in Equation 2. Suppose that at a later time we decode using the firing rates given by Equation 5 instead. The decoding methods we use work for a wide variety of tuning curve shapes, so such a switch is possible even if the tuning curves are distorted from their original form. The result of such a modified decoding scheme would be a population vector that is identical to the sensory input vector. Switching decoding schemes removes the experienced-based shifts in the population vector that reflect the training sequence. When translated into a more realistic framework, this provides a simple mechanism for erasing all traces of previously learned sequences without modifying any synapses within the coding array.

Within the nervous system, decoding of the type we use is unlikely to occur. Rather, information is transferred from one network to another in a distributed form. It has been shown that synaptic connections that accurately transfer encoded information from one network to another can arise from a correlation-based learning rule (Salinas and Abbott, 1995). Furthermore, there is a direct correspondence between the pattern of synaptic connections needed to transfer information and the decoding scheme that reveals the information being transferred. Suppose that the firing rates of the coding array were characterized by tuning curves \( f \) during the time that the synaptic connections from the coding array to a downstream network were established. The value encoded in the downstream array will then be the same as the value obtained from our decoding scheme using the tuning curves \( f \) (Salinas and Abbott, 1995).

An implicit assumption in our work is that synaptic connections from the coding array to downstream networks are not changed during the training experience that modifies synapses within the coding array. As a result, the information transferred to downstream networks after training is revealed by decoding the output of the coding array using the pretraining tuning curves, as we have done. However, synapses to downstream networks could be modified at a later time. Suppose that the same processes that originally established the synaptic weights from the coding array to a downstream network is activated sometime after training. The new synapses that develop between these two networks will no longer transfer the value decoded using the original tuning curves \( f \). Instead, the relevant tuning curves will be those given by Equation 5. Once the synapses to the downstream networks have been modified, the value transferred to them and revealed by the population vector will be identical to the sensory input. The previous training-induced shifts will have been completely eliminated. Thus, modifying and resetting synapses to downstream networks completely erases all information about previously learned sequences, leaving the network in a state suitable for learning new and different sequences.

The mechanisms for readout and erasure of stored sequences just outlined provide a rather novel picture of what a memory is. The sequence memories we have studied are relative; they are recalled through the interaction of a representation that was modified by training with another unmodified representation. The memorized information resides in differences between these two representations and memory is erased if both are modified so that these differences are eliminated. The pattern of synaptic weights within the coding array is, by itself, meaningless, since it can be canceled by a set of corresponding weights between the coding and readout networks. In this view, information is stored and recalled only through relationships between two sets of synaptic weights, in particular, through experience-induced differences between them.

**Experimental Predictions**

Our model makes some specific prediction that can be tested experimentally. First, we predict that the responses of encoding neurons will be shifted by training experience in a distinct manner. Equations 4 and 5 provide this prediction, which is shown, for one particular example, in Figure 5. Figure 5 shows the effect of the training sequence used in Figure 1a on the firing rates of four individual encoding neurons. The most important prediction is that neurons with tuning curves overlapping a training sequence will be elongated along the direction of the training sequence and shifted backward along the path of the training sequence. This effect is intimately related to the tendency of the population vector to lead the sequence. Experimental verification of this effect would be strong evidence for the presence of a predicting representation. The parameters chosen for Figure 5 resulted in a small amount of elongation and a larger backward shift. However, this parameter choice is not unique so we would more generally predict two consequences of training: (1) an elongation of the receptive field along the path of the training sequence, especially in cases where the sequence is experienced in both directions during training. (2) An additional backward shift of the receptive field when the sequence is experienced unidirectionally during training. Finally, the re-
ceptive fields shown in Figure 5 are larger after training than before. This is because our model uses only LTP so that firing rates are increased by training. If LTD or enhancement of inhibition also occurs during training, the growth of the receptive fields may not be as pronounced.

It may be possible to generate maps like those in Figure 1 directly from experimental data. Equation 6 or 26 allows the arrows in these maps to be computed if the tuning curves are measured and if the training-induced shifts in synaptic efficacy, represented by $\Delta W_p$, are known. Unfortunately, it is not possible to measure large numbers of synaptic weights directly, but it may be possible to infer them from firing correlation measurements (Georgopoulos et al., 1993; Wilson and McNaughton, 1994). If the asymmetric, short-latency firing correlation matrix is used in place of $\Delta W_p$ in Equation 6 or (5.11), a map representing the effect of training on a neuronal representation of sensory data can be generated. Results like those in Figure 1 would provide remarkable insight into the functional significance of changes in synaptic efficacy.

Generating Motor Sequences
In our model for the generation of learned motor sequences, proprioceptive feedback provides the input that drives further movement (Feldman, 1966; Ghez et al., 1990). Proprioception is known to play a key role in movement generation in humans (Sittig et al., 1987; Soechting and Flanders, 1989; Cordo, 1990; Cordo et al., 1994), other mammals (Andersson and Grillner, 1983; Conway et al., 1987; Saling et al., 1992) and insects (Bassler, 1986; Wolf and Pearson, 1988). Attempts to generate movements in the absence of proprioceptive feedback have been studied in patients with an unusual neuropathy (Rothwell et al., 1982; Sanes et al., 1985; Cole and Sedgwick, 1992). Psychophysical results suggest that during movement the perception of a limb position may lead its actual location (Dassonville, 1995) as we would predict. During rapid movements, it is likely that limb velocity rather than limb position is the correct encoded quantity to consider (Sittig et al., 1987; Schwartz, 1994). For very rapid movements, proprioceptive feedback about limb position may be too delayed to generate the population vector guiding movement rapidly enough. It has been suggested that the cerebellum may generate an internal copy of the proprioceptive feedback (Miall et al., 1993) and it might be possible that such an internally generated proprioceptive signal could be used to drive recall of learned movements requiring very rapid feedback.

A number of models of arm movements have been developed (Flash, 1987; Bullock and Grossberg, 1988; Kawato et al., 1988; Kuperstein, 1988; Massone and Bizzchi, 1989; Uno et al., 1990; Houk et al., 1990; Jeannerod, 1990; Lukashin, 1990; Guldiano and Grossberg, 1991; Burnod et al., 1992; Musa-Ivaldi and Giszter, 1992; Berthier et al., 1993; Lukashin and Georgopoulos, 1993). Many of these are complementary to our model (although see Houk et al., 1993; Berthier et al., 1993; Lukashin and Georgopoulos, 1993), but some of the mechanisms discussed in these models can be used in our approach to transfer information from the proprioceptive array that provides the target location of a learned sequence to motor circuits (Salinas and Abbott, 1995).

Schemes for generating sequences have been constructed using mathematical neural network models (see Amit, 1989; Hertz et al., 1991; Minai and Levy, 1993). A key distinction between these models and ours concerns the rate at which the learned sequence is generated. In previous models, nonlinear networks produce learned sequences autonomously through internal feedback. This means that a sequence is generated at a speed characteristic of neuronal dynamics. In most cases, this is far too fast to direct a motor system. Furthermore, it does not allow for any adjustments to delays or errors in the motor response. Thus, even if the sequence could be generated slowly enough, the movement could only be performed at one speed and it would not be stable to perturbations. By using population decoding techniques, we have been able to include proprioceptive feedback from the moving limb into our model and we use it, not internal feedback, to drive the network. Since recall is response driven, the learned sequence is generated at a rate that matches the task. Furthermore, once it is learned, the task can be performed at any desired speed with or without pauses and stability of the motion is assured.

Appendix A
Equation 6 is derived by substituting Equation 5 for the firing rates after training into Equation 1 defining the population vector giving

$$\tilde{p} = \frac{\sum_i f_i(\tilde{x}) + \sum_i \Delta W_i f_i(\tilde{x})}{\sum_i f_i(\tilde{x}) + \sum_i \Delta W_i f_i(\tilde{x})^2}.$$  (7)

We assume that before training, when $\Delta W = 0$, the population vector is equal to the sensory input so

$$\sum_i f_i(\tilde{x}) \tilde{x}_i - \tilde{x} = 0.$$  (8)

Equation 6 follows from this and an expansion of the denominator of Equation 7 to linear order in $\Delta W$.

The analogous result for the shift in the population vector using the least-squares method is obtained by substituting Equation 5 into Equation 2 so that $\tilde{p}$ is determined by the condition

$$\sum_i f_i(\tilde{x}) + \sum_i \Delta W_i f_i(\tilde{x}) - f_i(\tilde{p}) = 0.$$  (9)

It is convenient to express the population vector as

$$\tilde{p} = \tilde{x} + \Delta \tilde{p}.$$  (10)

and we compute to linear order in $\Delta \tilde{p}$ writing

$$f_i(\tilde{p}) = f_i(\tilde{x}) + \tilde{p}_i f_i(\tilde{x}) \cdot \Delta \tilde{p}.$$  (11)

Performing the minimization with respect to $\Delta \tilde{p}$ we find

$$\sum_i \left( \Delta W_i f_i(\tilde{x}) - \tilde{p}_i f_i(\tilde{x}) \cdot \Delta \tilde{p} \right) \tilde{p}_i f_i(\tilde{x}) = 0.$$  (12)

Defining the matrix

$$Q = \sum_i \tilde{p}_i f_i(\tilde{x}) \tilde{p}_i f_i(\tilde{x}),$$  (13)

the solution is

$$\Delta \tilde{p} = Q^{-1} \sum_i \tilde{p}_i f_i(\tilde{x}) \Delta W_i f_i(\tilde{x}).$$  (14)

If the tuning curves $f$ are uniformly arranged the inverse matrix just introduces a numerical constant and

$$\Delta \tilde{p} = \sum_i \tilde{p}_i f_i(\tilde{x}) \Delta W_i f_i(\tilde{x}).$$  (15)

Appendix B
Equations 4 and 6 provide the results needed to generate the figures shown. We will discuss how these equations are evaluated for the case when $\tilde{x}$ is two-dimensional as in the figures.
Other dimensions involve trivial changes in the formulas. The tuning curves are taken to be Gaussian,

\[ f_s(\mathbf{x}) = R_{\text{max}} \exp \left( -\frac{||\mathbf{x} - \mathbf{\bar{x}}||^2}{2\sigma^2} \right), \]  

and the vector \( \mathbf{\bar{x}} \) is allowed to extend over an infinite range. We assume that the maximal response values \( \mathbf{\bar{s}} \) are spread uniformly and that the firing rate curves are highly overlapping. This allows us to replace sums over coding neurons with integrals over their maximal response vectors,

\[ \sum_{\mathbf{s}} \rightarrow \rho \int d\mathbf{\bar{s}} \]  

where \( \rho \) is equal to the number of neurons with \( \mathbf{\bar{s}} \) vectors lying within one unit area in the space of \( \mathbf{\bar{x}} \) values. We have checked the accuracy of this approximation and found it to be very good even for a modest number of neurons (Salinas and Abbott, unpublished observations). For simplicity, when computing the effects of LTP, we assume all-to-all coupling between coding neurons. Using these approximations, Equation 6 becomes

\[ \mathbf{\bar{p}} = \mathbf{\bar{x}} + \frac{\mathbf{\bar{A}}}{\rho} \]  

where

\[ \mathbf{\bar{A}} = \rho^2 R_{\text{max}} \int d\mathbf{\bar{s}}_1 d\mathbf{\bar{s}}_2 (\mathbf{\bar{s}}_1 - \mathbf{\bar{x}}) \cdot \Delta W(\mathbf{\bar{s}}_1 - \mathbf{\bar{s}}_2) \exp \left( -\frac{||\mathbf{x} - \mathbf{\bar{z}}||^2}{2\sigma^2} \right) \]  

and

\[ B = \rho R_{\text{max}} \int d\mathbf{\bar{s}} \exp \left( -\frac{||\mathbf{x} - \mathbf{\bar{z}}||^2}{2\sigma^2} \right) \]  

Performing the Gaussian integrals gives

\[ \mathbf{\bar{p}} = \mathbf{\bar{x}} + \pi \sigma^2 \rho R_{\text{max}} \int dt dt' H(t') \exp \left( -\frac{||\mathbf{\bar{X}}(t + t') - \mathbf{\bar{X}}(t) + \mathbf{\bar{X}}(t')||^2}{2\sigma^2} \right) \]  

where \( \mathbf{\bar{X}}' = d\mathbf{\bar{X}}/dt \). With this approximation we find

\[ \mathbf{\bar{p}} = \mathbf{\bar{x}} + \pi b R_{\text{max}} \rho^2 \int dt \mathbf{\bar{X}}(t) - \mathbf{\bar{x}} + \mathbf{\bar{X}}'(t) \exp \left( -\frac{||\mathbf{\bar{X}}(t) - \mathbf{\bar{X}}'||^2}{4\sigma^2} \right). \]

Here \( b \) is the time integral of \( H \) and \( \tau \) is average LTP window time given by

\[ \tau = \frac{1}{b} \int dt dt' H(t'). \]

The term \( \mathbf{\bar{X}} - \mathbf{\bar{x}} \) in the integral of Equation 24 pulls the population vector toward values in the training sequence while the term proportional to \( \mathbf{\bar{X}}' \) moves it forward along the training sequence providing future prediction. The figures were generated by numerically integrating Equation 24 using \( \pi b R_{\text{max}} \rho^2 = 2, \tau = 200 \text{ms} \), and \( \sigma = 0.7 \). The magnitude of the velocity \( ||\mathbf{\bar{X}}'|| \) was 5/s except for the circular path where it was 12.5/s.

For the case of the least-square method, we obtain similar results. We evaluate Equation 14 for the training-induced shift in the population vector exactly as above using Gaussian tuning curves and replacing sums by integrals. The results are

\[ \Delta \mathbf{\bar{p}} = \frac{2b}{\pi} \rho \sum_{\mathbf{\bar{s}}} \mathbf{\bar{V}}(\mathbf{\bar{s}}) \Delta W_{\mathbf{\bar{s}}} \]  

and

\[ \Delta \mathbf{\bar{p}} = \pi b R_{\text{max}} \rho^2 \int dt \frac{\mathbf{\bar{X}}(t) + \mathbf{\bar{X}}'(t) \cdot \mathbf{\bar{X}}(t) - \mathbf{\bar{x}}(t) ||^2}{2\sigma^2} \]  

where \( \mathbf{\bar{x}}(t) = \mathbf{\bar{X}}(t) - \mathbf{\bar{x}} \). This \( \Delta \mathbf{\bar{p}} \) is not identical to the difference \( \mathbf{\bar{p}} - \mathbf{\bar{x}} \) from Equation 24, but it produces very similar figures. If diagrams like those of Figure 1 are generated using this result, the arrows beside the training sequence point slightly more in the direction toward the training path and slightly less along it. For points on the training path, the direction of the arrows is unchanged, but they are slightly shorter for the same parameter values.

It is interesting to evaluate Equation 24 for the case where sensory inputs in the training sequence change at a constant "velocity," \( \mathbf{\bar{V}}(t) = \dot{\mathbf{\bar{V}}} \). This provides a good approximation of the population vector for sensory inputs in a sufficiently small range near the middle of a long training sequence. In this case,

\[ \mathbf{\bar{p}} = \mathbf{\bar{x}} + 2 \pi \sigma^2 b R_{\text{max}} \rho^2 \int \mathbf{\bar{V}}(t) \cdot \dot{\mathbf{\bar{V}}} \exp \left( -\frac{||\mathbf{\bar{X}}(t) - \mathbf{\bar{V}}||^2}{4\sigma^2} \right) \]  

where \( \mathbf{\bar{V}} \) is the component of \( \mathbf{\bar{x}} \) perpendicular to the path of the training sequence, \( \mathbf{\bar{X}}_{\perp} = \mathbf{\bar{x}} - \mathbf{\bar{V}} \cdot \mathbf{\bar{V}} \) and \( \mathbf{\bar{V}} = \mathbf{\bar{V}}/||\mathbf{\bar{V}}|| \). The term proportional to \( \dot{\mathbf{\bar{V}}} \) moves the population vector forward along the training sequence. The fact that this term is proportional to \( \dot{\mathbf{\bar{V}}} \) means that it is independent of the magnitude of the training "velocity" vector. The power to predict decreases exponentially with the square of the distance between the sensory input \( \mathbf{\bar{x}} \) and the training sequence. In Equation 28, \( \pi b R_{\text{max}} \sigma \tau \) is the amount of synaptic potentiation that occurs in one LTP window time between two neurons firing at rates \( R_{\text{max}} \). The factor \( \rho^2 \) is equal to the number of units with maximal response vectors \( \mathbf{\bar{s}} \) lying within one \( \sigma \) area. This is equal to the number of neurons with tuning curves that peak within the width of a typical tuning curve and is a measure of the degree of overlap between the coding neurons. The term \( \sigma/||\mathbf{\bar{V}}|| \) is approximately the length of time that any one neuron fires during the training sequence.

Notes
This work was supported by NSF-DMS9208206 and the McDonnell-Pew Centre for Cognitive Neuroscience at Oxford (L.A.) and NIH-NS07292 (K.B.). We thank Marco Idiart, John Iisman, and Eve Marder for discussions.

Address correspondence to L. F. Abbott, Center for Complex Systems, Brandeis University, Waltham, MA 02254.
References


Linsker R (1986) From basic network principles to neural architecture.


Salinas E, Abbott LF (1995) Transfer of information between sensory


Rothwell JC, Traub MM, Day BL, Obeso JA, Thomas PK, Marsden CD

Rolls ET (1989) Functions of neuronal networks in the hippocampus

O'Neill WE, Suga N (1982) Encoding of target range information and


O'Keefe J, Dostrovsky J (1971) The hippocampus as a spatial map:

Mussa-Ivaldi FA, Giszter SF (1992) Vector-field approximation: a com-


