Simple Neural Circuits and Linear Algebra

Perhaps the simplest possible neural circuit consists of a single postsynaptic neuron (which we will label I) together with the set of presynaptic neurons that innervate it (which we will label J1, J2, etc). An obvious question is, how does the sequence of spikes emitted by I depend on the sequences of spikes in the J cells? A second obvious question is, since the transformation from the presynaptic spikes to the postsynaptic spikes is essentially the “computation” or “information processing” done by this circuit, how can it be ensured that this transformation is actually useful to the organism?
The first question is one of mechanism, and the second one of purpose, but at some level the answers to the 2 questions are interrelated, in the same way that the questions “how does the kidney produce urine?” and “why is excretion useful?” are interrelated.

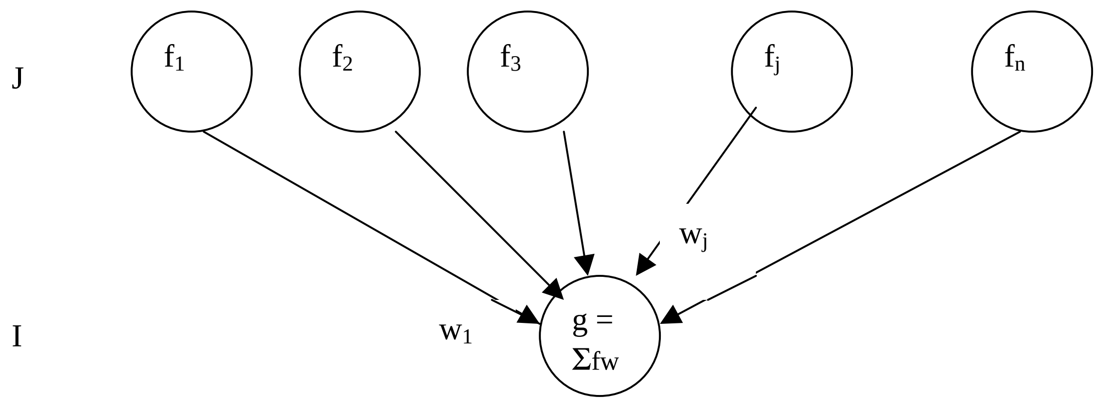

Fig 1. The Ideal (Connectionist) Neuron. Output neuron I (bottom circle) gets synaptic inputs from a set of n input neurons J (top circles). The synaptic strengths (or “weights”) are labeled w_j etc (arrows). The firing rate of the jth input neuron is f_j, and the firing rate of the output neuron, g, is given by the weighted sum of the input firing rates.

From your knowledge of basic cellular neurophysiology, you can see that the answer to the first question is in principle straightforward, though exceedingly complex. One would need to know how the presynaptic spikes travel down the various axons (Hodgkin-Huxley equations), how these arriving spikes affect transmitter release (time course of I_{Ca}, Ca diffusion and binding to calcium sensors, effects on statistical parameters like p and n, cleft and postsynaptic processes like those we considered for mini generation at the neuromuscular junction, conversion of channel opening (synaptic conductances changes) to local voltage changes, spreading of local voltages along dendrites (using cable theory but also considering the possibility of voltage-dependent dendritic conductances), and finally initiation of spikes in the initial segment of I. Fortunately, from our knowledge of these constituent events, we can make many reasonable approximations and simplifications. Perhaps the most drastic simplification of all would be

(1) **Rate Assumption**. Represent the activity of each presynaptic neuron not by its detailed spike history, but a single number, its average firing frequency f_j (j is an "index" that indicates which of the J cells we are considering; thus for cell J1, j = 1 etc).
(2) **Weight Assumption** Assume that a given presynaptic neuron firing at a rate f_j generates a current at the initial segment of neuron I which is proportional to the product of f_j and the total "strength" of the synapses that cell J makes on cell I. This strength plays the role of a "weight" in a weighted sum (see below) so we represent it quantitatively by the symbol w_j . The strengths incorporate factors such as the number and location of synapses, the amount of transmitter released, and the number of postsynaptic receptors.
(3) **Summation Assumption**. Assume that all the currents arriving at the initial segment add together linearly, so the total current is proportional to $\Sigma f_j w_j$ which stands for " the sum of all the pairwise products of the presynaptic firing rates and the corresponding presynaptic strengths" (see below)
(4) **Linearity Assumption**. Assume that the firing rate of the postsynaptic cell, g, is proportional to this total current.

These assumptions can combined in the following equation:

$$g = \sum_{j=1}^{n} f_j w_j \quad \text{.....................................Eq 1}$$

which can be written out in full as

$$g = f_1w_1 + f_2w_2 + f_3w_3 \ldots\ldots\ldots + f_nw_n \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\text{..Eq 2}$$

where there is a product on the right for each of the total of n presynaptic cells. Any proportionality constants have been incorporated into the units of w.

We will now discuss these assumptions in more detail, to make sure they are reasonable, and to see how they might be made more realistic if we were willing to use a more elaborate equation. All the assumptions are rather drastic, but nevertheless the idealized minicircuit that results has quite interesting computational properties, and captures some of the key features of real neurons. Our assumptions, and their mathematical formulation, constitute a "model". Later on we will consider more elaborate models.

Rate Assumption . We represent the output of a neuron as a single number, its " spike frequency" or rate. This is often referred to as a rate-based model, or simply a rate-model. Essentially we are assuming that the precise temporal pattern of the spikes does not matter – the spikes could be arriving in a sudden burst, or very regularly, or randomly (in which case the intervals between the spikes would be exponentially distributed a la Poisson, except for very short intervals comparable to the refractory period). This would be true if the time window over which neurons temporally sum their inputs (essentially the membrane time constant) was very long compared to the times between spikes. Since neurons can fire at very low frequencies, this assumption appears at first glance to be untenable. However, since in many cases the postsynaptic neuron receives input from a large number of presynaptic cells, spikes will be typically arriving at quite a high rate, even though some presynaptic cells are firing slowly. Furthermore, if many presynaptic cells must cooperate to fire a postsynaptic cell, then most of the temporal details of the incoming spike trains will get washed out (unless of course there are temporal correlations in the firings of the input cells). However, there is good evidence that in some sensory systems the precise timing of individual spikes is important.

Weight Assumption . Because we neglect the arrival times of spikes, we are picturing the synapses as releasing transmitter at a constant rate, which depends on the spike frequency. This leads to opening of postsynaptic ion channels, whose number would depend linearly on the presynaptic firing rate. Thus we are neglecting possible postsynaptic receptor saturation, the requirement for 2 transmitter molecules per receptor, as well as the complex diffusion and binding processes we considered at the neuromuscular junction. However, if each connection (i.e. the set of synapses formed from a given presynaptic neuron onto the postsynaptic neuron) is comprised of many synapses (as at the neuromuscular junction) and these synapses have fairly low probability of release and/or are weak, the net synaptic current would be approximately proportional to the presynaptic firing rate and to the aggregate synaptic strength, as long as the synaptic conductance changes were small compared to the input conductance of the cell ($G_{syn} << G_{in}$). Many synapses are formed on narrow dendrites, which will have small values of input conductance; the resulting large local synaptic depolarization may approach the reversal potential; the resulting nonlinearity could be offset by having synapses scattered all over the dendritic tree rather than being concentrated at one location – this is often seen.

Summation Assumption . All the synaptic currents induced at each connection add together linearly at the level of the initial segment. The combined depolarization must be much less than the driving force ($V_m – E_{syn}$); furthermore the current must flow passively along the dendrite to the cell body. In reality it is likely that there are active voltage-dependent dendritic channels. However there is some evidence that the effect of these active conductances is to compensate for the attenuation and temporal distortion introduced by the cable properties of the dendrites, so that the overall transfer of charge from synapse to initial segment is surprisingly faithful.

Linearity Assumption . If the H-H equations are supplemented by A-currents then computer calculations suggest that the relation between firing rate and injected current can be quite linear, at least above threshold and at frequencies below the reciprocal of the refractory period. Another way to approach this issue is to consider a highly simplified picture of spike firing called Leaky Integrator Firing. In this model, the cell is considered to be purely passive, except that whenever a fixed voltage threshold is reached the cell fires an infinitely short spike, following which the membrane potential resets to rest. Following a refractory period τ_{ref} the membrane potential then passively depolarizes again (in response to the applied current I). It can be shown [see Koch pg 338] that the firing frequency for this model is given by $1/f = \tau_{ref} - \tau_m \ln (1-I_t/I)$ where I_t is the threshold current. For short τ_{ref} and large I this becomes simply $f = I/\tau_m I_t$ i.e. the postulated linear relation.

Returning to our basic linear model, Eq 1 or 2 , we see that the postsynaptic firing rate is given as a linear sum of the weighted presynaptic firing rates, the weights being the synaptic strengths (the units in which the strengths are measured incorporate various proportionality constants). This idealized picture of a real neuron is sometimes known as the generic "connectionist" neuron, because the essence of the computation performed is contained in the pattern of connections it receives, rather than the details of the biophysical properties of the neurons themselves. (However, connectionist neurons often assume a nonlinear f-I assumption).

At this point we should consider whether the values of f and w can be allowed to take negative values. At first glance this makes little sense: neurons cannot have negative firing rates, and shunting (chloride dependent) inhibition does not inject hyperpolarizing current. However, mathematical analysis is made easier by allowing negative numbers. Furthermore, we can imagine that neurons with negative firing rates are in fact neurons with positive firing rates whose excitatory synapses have all been made inhibitory (and vice versa). In the real brain, both negative and positive quantities (eg shade and light) are represented as the positive firing rates of 2 sorts of neurons working in "push-pull": one cell carries the positive part of the signal, and another the negative part. So our idealized neuron can be thought of as combining a pair of more realistic neurons. Also, even though we described GABA-A inhibition as primarily due to shunting, the actual effect, in both H-H neurons and LIF neurons, is close to subtractive (i.e. linear). This arises because while shunting decreases τ_m it increases I_t by the same amount. The model also violates "Dale's Law" – that one neuron uses 1 transmitter, such as glutamate or GABA, and is therefore unlikely to make both excitatory and inhibitory synapses. Again our ideal neuron collapses both types of neuron, which in the brain are distinct, into one.

What significance does Eq 1 (or Eq 2) have? The brain is primarily interested in patterns, that is, sets of numbers (brightnesses, pressures, firing rates, etc). The set of presynaptic firing rates f_1, f_2, etc constitutes a pattern. For example, this could be the firing rates of photoreceptors in the eye that are responding to a particular pattern of light (i.e. an image). In this case the pattern f_1,f_2,etc would be the various pixel values across an image. In mathematics we call such a pattern (i.e. an ordered set of numbers) a *vector*. So the set of firing rates f_1, f_2 etc is a vector. A convenient notation for the set of numbers $f_1,f_2,\ldots$ is the symbol **f** (note the bold face). **f** is not a single number, it is an entire set of numbers. The components of **f** are $f_1,f_2,f_3\ldots$etc all the way up to f_n (assuming there are n presynaptic neurons altogether). We say that the dimensionality of **f** is n. The boldface warns us that **f** cannot be treated as a single number. But if we multiply each component of **f** by a (where a is some number), then the result can be written a**f** (meaning $af_1 + af_2 \ldots\ldots +af_n$**).** Also, the sum of 2 vectors **f** and **w** is the vector $f_1+g_1,f_2+g_2\ldots\ldots$

Now in equation 1 (or 2) we have two sets of patterns, or vectors : $f_1, f_2 \ldots .. f_n$ and $w_1, w_2, \ldots w_n$. The first vector is the firing (or activity) vector **v** and the second vector is the weight (or strength) vector **w**. We call the weighted sum $\Sigma\, f_j\, w_j$ the "dot product" or "inner product' of **f** and **w** (since it is the sum of the *products* of the individual components of the vectors). The dot product of 2 vectors **f** and **w** is written $\mathbf{f} \cdot \mathbf{w}$. The idea that a neuron may compute the dot product of its input vector and its weight vector is one of the most important ideas in neuroscience.

[Note on notation. A vector is an ordered set of numbers, which we can write either as a column of numbers or a row of numbers. This leads to the notation **f** for a row vector and $\mathbf{f}^T$ for a column vector (^{T} indicates transposition). We use the convention that $\mathbf{f}^T\mathbf{g} = \mathbf{f.g}$ (i.e. that a dot product is a column vector times a row vector). The reverse order, $\mathbf{gf}^\mathbf{T}$**,** defines the so-called "outer product", which is a matrix **X** whose element in column j and row i, $x_{j,i}$, is given by the jth element of **f** and the ith element of **g.**]

Why is this idea so important? It is because the dot product is a simple measure of the similarity of 2 vectors, and since the brain is primarily interested in patterns, and their similarities, the dot product operation becomes vital. However, we have to be a bit careful about comparing patterns of numbers. In one sense the patterns 1,2,3 and 3,6,9 are very similar : they are scaled versions of each other , the scaling factor being 3. In another sense 1,2,3 is quite similar to 2,2,2 (they are all small numbers). The first similarity is like that between dim and bright pictures of a cow, the second similarity is like that between a dim picture of a cow and a dim picture of a tiger. It is usually more important in life to be able to distinguish a cow from a tiger than a cow in shadow from a cow in sunlight. We can make this more precise by thinking about the *direction* and *length* of vectors. Let us start by considering a 2 dimensional vector $\mathbf{f}^1$ with components f_1^1 and f_2^1. We can plot this vector as a point on a graph whose abscissa (x-axis) represents f_1 and whose ordinate (y-axis) represents f_2. However the vector is actually the relationship between this point and the origin, i.e. the line between f_1^1, f_2^1 and 0,0. This line has both direction and length. Now on the same graph place a second vector $\mathbf{f}^2$ (notice that the superscripts here do not imply "raising to a power": they are simply distinguishing labels on our 2 vectors). If $\mathbf{f}^2 = a\, \mathbf{f}^1$ (i.e. just a scaled version), then the 2 points will lie along the same line, i.e. they share *direction*. However, if the relative values of the components of each vector are different, they have different directions, even if the lengths of the lines linking the points to the origin (the lengths of the vectors) are identical. In 2 dimensions the lengths of the vectors are equal to the hypotenuses of the right triangles whose other sides' lengths are the vector components. These vector lengths are calculated using Pythagoras's theorem.

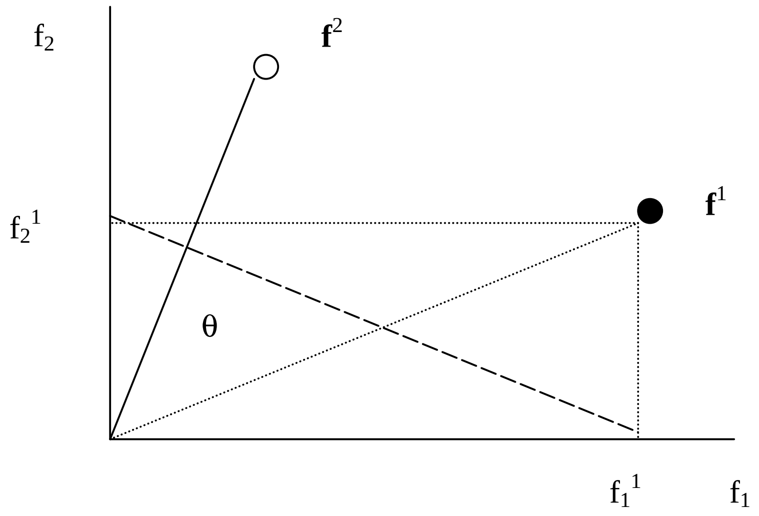

Fig 2. Visualising Vectors. 2 dimensional vectors, which have just pairs of elements (or components), can be drawn on paper. For example the vector $\mathbf{f}^1$, with components f_1^1 and f_2^1, is the dotted line connecting the filled circle to the origin. The vector $\mathbf{f}^2$ is the line connecting the open circle to the origin. These 2 vectors differ in both length and direction. The angle between the vectors in θ. If the vectors are just scaled versions of each other, the angle is zero. Note the length of vector f^1 is equal to that of the dashed line, which is the hypotenuse of the right triangle formed by $0\text{-}f_1^1$ and $0\text{-}f_2^1$. It is thus given by the square root of $(f_1^1 f_1^1) + (f_2^1 f_2^1)$

We would like our measure of similarity to be 1 when the 2 vectors have the same direction, and 0 when they lie at ninety degrees to each other (that is, they are *orthogonal*). In other words, we want our measure of similarity to be 1 when the angle between the vectors (denoted θ) is zero, and to be 0 when the angle is 90 degrees. This is achieved by using the cosine of θ ($\cos 0^o = 1$, $\cos 90^o = 0$). Now it can be shown that for the 2 vectors **f** and **w** separated by the angle θ

$$\cos\theta = \mathbf{f} \cdot \mathbf{w} \,/\, (\mathbf{f} \cdot \mathbf{f})^{1/2} (\mathbf{w} \cdot \mathbf{w})^{1/2} \quad \text{……………………………..Eq 3}$$

where $(\mathbf{f} \cdot \mathbf{f})^{1/2}$ (i.e. the square root of the dot product of **f** with itself) is the length of the vector **f**, and similarly for **w**. (These lengths result from extending Pythagoras's theorem to higher dimensions).

Thus, provided we are comparing 2 vectors of the same length (preferably unit length), their dot product provides a natural measure of similarity. In a sense, the basic computation performed by the generic connectionist neuron is a measurement of the similarity between its current input pattern **f** and its stored reference pattern **w**. Although directions and lengths are most easily depicted in 2 dimensions, and can only be visualized up to 3 dimensions, all these concepts have validity in many dimensions. Patterns are just vectors in high dimensional space, but they still point in specific directions in that space, and have specific lengths. Images are vectors whose components are the pixels comprising the image. Smells are vectors which are represented in the brain by the relative activity of the 1000 different glomeruli in the olfactory bulb. A musical chord is a vector whose elements are the loudnesses of the notes from top to bottom of the keyboard.

One more concept we need, already touched on above, is that of orthogonality. If 2 patterns are as different as possible, we say that they are orthogonal (because the corresponding vectors are at right angles). A simple example would be 2 DNA sequences which differ in every position. (The number of positions in which each base is the same, divided by the total number of bases, is just cos θ; here the length of the vector is just the chain length). The dot product of 2 orthogonal vectors is zero whatever their length (look at Eq 3: if $\theta = 90^o$, $\cos \theta = 0$).

We can illustrate these ideas using a very simple memory model. A key idea in neuroscience, which we will explore much more, is that learning occurs by changing the strengths of synaptic connections (although changes in the firing properties of neurons may also be important) as a result of neural activity. These connections could be changed by some external agency, such as an omniscient homunculus deep inside the brain, which would be able to visit each synapse and regulated its strength according to some pre-ordained recipe. In a way, such a genie does exist: our genomes. We saw in the olfactory system that elaborate and highly specific patterns of connections could be laid down by a set of chemoaffinity labels, the odorant receptor proteins, in the absence of neural signals. However, this approach to wiring already consumed some 3% of the mammalian genome just for the first set of neurons alone. It seems impossible that the whole brain could be wired up in this way, even if this were desirable. However, it is not desirable since it would mean that everyone’s brain would work the same, even though their circumstances and experience are completely different. Such “hardwiring” is only suitable when environmental regularities persist over many thousands of generations. Even for regularities that have persisted for aeons, such as natural scenery and animal behaviors, these regularities are so complex that it is probably impossible to encode them using a rather small genome.

So neuroscientists believe that the strengths of synapses are controlled by their past history of electrical activity. Indeed the strengths of synapses must be controlled, not by the global pattern of activity across a set of neurons, but by the activity that actually reaches the synapse (i.e. the pre- and postsynaptic spikes, possibly combined with some general neuromodulatory influences). We call this *local* learning.

In the case of a single postsynaptic neuron receiving inputs from a set of presynaptic cells, the simplest way for the synaptic strengths to change would be as a consequence of the arrival of the presynaptic spikes themselves at the synapses. This would be a use-dependent synaptic learning rule, rather like the strengthening of muscles according to how much exercise they receive. This learning rule could be written (Δ stands for “the change in…”

$\Delta w_j = f_j$ …………….Eq 4

We could incorporate this into our model as follows. Let us try to learn a set of m input patterns, $\mathbf{f}^1, \mathbf{f}^2, \ldots \mathbf{f}^k \ldots \mathbf{f}^m$ (note that each pattern is a vector comprised of the activities f_1, f_2 etc; the superscripts label the patterns – they do NOT mean that we are raising the vector to a power, which is a meaningless idea ; we will assume that all the input vectors are orthogonal and have unit length; also the average value of any element across all the patterns is assumed to be zero – on a computer screen this would be represented as gray, and we are assuming that a pixel is as likely to be black as it is to be white, which over a sufficient set of images is likely to be true). If we start with all the strengths set at zero, then the first pattern on the inputs would cause the strength of the J1-I connection to increase to a level f_1^1 (i.e. the weight w_1 increases to the value given by the firing rate of the first presynaptic neuron). Similarly w_2 increases to f_2^1 and all the elements of $\mathbf{f}^1$ get "imprinted" on the strength vector. The next pattern $\mathbf{f}^2$ produces a further increase in weight (actually, since the elements of the vectors can take negative values, the strengths could also decrease). The final set of weights resulting from learning all m patterns will be given by

$$\mathbf{w} = \sum_{k=1}^{m} \mathbf{f}^k \text{……………………eq 5}$$

where $\mathbf{f}^k$ refers to the kth pattern. The sum is over all the m patterns (i.e. the final weight vector is given by the sum of the changes due to all the individual patterns). We are adding together corresponding elements of all the input vectors.

What will be the output of the postsynaptic neuron if one of the learned patterns (say pattern number 1) is provided as a test input ? It will be given by

$$g^k = \mathbf{w} \cdot \mathbf{f}^1 = \sum_{k=1}^{m} \mathbf{f}^k \cdot \mathbf{f}^1 = \mathbf{f}^1 \cdot \mathbf{f}^1 + \sum_{k \text{ not } 1}^{m} \mathbf{f}^k \cdot \mathbf{f}^1 \qquad \text{…………..eq 6}$$

What we have done here is first to define the output in terms of the dot product of the current input vector $\mathbf{f}^1$and the learned weight vector **w,** which we then replaced by the sum of all the previously learned patterns $\Sigma\, \mathbf{f}^k$, which is itself a vector. Now, the output is composed of 2 parts: a part $\mathbf{f}^1 \,.\, \mathbf{f}^1$ which is due to the dot product of the current pattern with the same but previously learned pattern (which was imprinted on the weights) and a part $\Sigma\, \mathbf{f}^k \,.\, \mathbf{f}^1$ (k not equal to 1) which reflects the sum of all the dot products of the current input pattern with all the previously learned patterns *except* the current pattern (which is expressed by the notation j is not equal to 1).

Now the term $\mathbf{f}^1 \cdot \mathbf{f}^1$ is just the length of the test input vector, which we assumed to be 1. Since all the patterns were assumed orthogonal the term $\Sigma\, \mathbf{f}_k \cdot \mathbf{f}^1$ must be zero. Thus our output neuron gives an output of one if it "sees" one of the stored patterns. However, if it is shown a new pattern, still orthogonal to all the stored patterns, it will give output zero. It signals that it "recognizes" the stored pattern but does not recognize the new pattern.

The pattern recognizer only works for orthogonal patterns, i.e. patterns that are very different. (A simple example would be 1111 and 1010, for polynucleotide sequences; see next lecture for discussion of orthogonality of polynucleotides). It recognizes perfectly up to *n* familiar orthogonal patterns.

What happens if the patterns are not orthogonal – that is, they are not as different from each other as possible? If the summed vector memory is shown a new pattern that resembles one of the remembered patterns, it will generate an output that is greater than 0 (because the second term in Eq 6 will no longer be exactly 0). However, if the dimensionality is very high then *random* patterns are almost orthogonal (provided the mean values of the elements are zero). This is because the dot products (which is proportional to cos θ) of any pair of such random vectors will get closer to zero as the dimensionality increases (although the individual products of corresponding elements will scatter around zero, their sums will approach zero more closely as the number of terms increases, and the standard deviation around zero of the dot products will fall as the reciprocal of the square root of the dimensionality).
We can assess the performance of the summed vector memory as a recognition device by computing its "signal to noise ratio" (S/N), defined as the average of the squares of the output when given stored items divided by the average squared output novel items. (We use the squares of the output to ensure that the the noise measurement is zero when shown novel items). It can be shown that for random patterns S/N = n/m (n is the dimensionality, i.e. the number of input patterns, and m is the number of stored patterns). The memory works best when it only stores a few patterns. It gets confused when trying to store too many patterns.
However, though the neuron signals pattern recognition, it does not allow retrieval of the pattern itself – obviously because the pattern itself contains a number of elements, whereas the output neuron itself is just one element. It acts as a familiarity meter, but not a real memory. (Note that often we can remember we have seen someone before, even though we can't remember their name; likewise we know we heard a tune before even though we cannot hum it). To do a better job, we need an entire set of p output neurons. Each of these output neurons should get input from each of the set of n input neurons, according to the idealized description above. So the output of the ith output cell would be given by

$g_i = \mathbf{f} \cdot \mathbf{w}_i$ ....eq 7

where $\mathbf{w}_\mathbf{i}$ is the vector of synaptic weights onto the ith output cell.

What will the vector of output activities **g** be given by? The ith element of this vector is given by the dot product of the input vector and the relevant weight vector $\mathbf{w}_i$. So the whole output vector will depend on the entire set of p weight vectors. We can list the elements of these weight vectors as the rows of a table of strengths $w_{i,j}$. The rows of the table correspond to the different postsynaptic cells labeled i. The columns of the table correspond to the different presynaptic cells labeled j. The weight $w_{i,j}$ is the strength of the connections from presynaptic neuron j to postsynaptic neuron i. Such a table of elements is called a matrix; we represent it using the symbol **W (**bold capitals are used for matrices). Using this notation we can write the output vector **g** as

g = Wf ……………..8

Note very carefully that none of the symbols **g**, **f** and **W** stand for single numbers. They stand for sets of numbers. This equation means that the ith element of **g** is calculated by taking the dot product of the ith row of numbers of the matrix **W** with the current input vector **f** (this dot product could be written as $\mathbf{f}^{\mathbf{T}}\mathbf{g}_i$, since **f** is a column vector and $\mathbf{g}^i$ is a row vector.)

So the synaptic matrix **W** (which represents the entire n by p set of connections from J cells to I cells) changes the input vector **f** to an output vector **g**. The actual output pattern we get depends on both the input vector and the weight matrix.

How can we interpret this process of transforming one vector into another using the set of rules contained in **W**? Remember that a vector is an oriented line in hyperspace. So the change wrought by W must correspond to twisting this line into a new direction and stretching (or shrinking) it to a new length. The effect of **W** depends on the vector it acts on: it could change its length, or direction, or both. Let's look at this in 2 dimensions, i.e using 2 component vectors. (Fig 3).

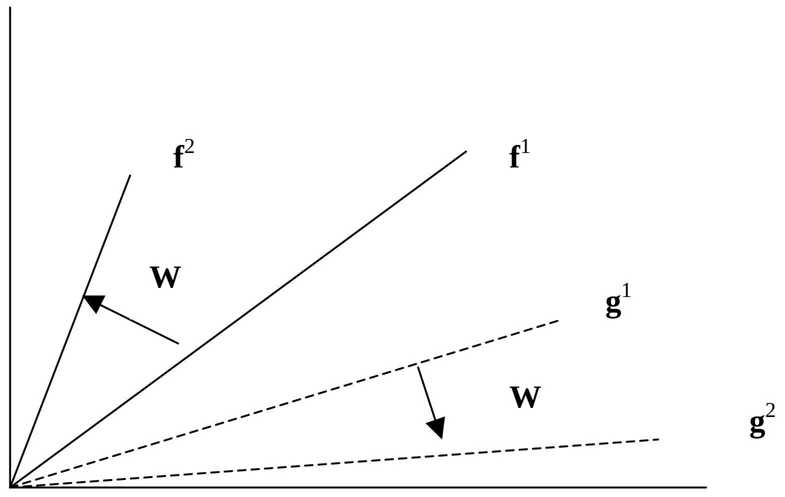

Fig 3. The matrix rotates and shortens the vector $\mathbf{f}^1$ into a new vector $\mathbf{f}^2$. The same matrix may rotate and stretch another vector $\mathbf{g}^1$ into $\mathbf{g}^2$. The illustration is just for 2-dimensional vectors, but remains true in higher dimensions.

Now we have a way of thinking about memory. We want to learn a set of weights **W** such that whenever we are presented with an input pattern **f** (e.g. a face) we will respond with an output pattern **g** (e.g. a name). This must be achieved by adjusting the individual weight $w_{i,j}$ using the information contained in **f** and **g**. To be neurobiologically realistic this means that the weight $w_{i,j}$ should be adjusted locally, that is, using the activity of the presynaptic neuron that forms that connection (i.e. f_j) and the activity of the postsynaptic neuron receiving the connection (i.e. g_i). How should f_j and g_i be combined to adjust $w_{i,j}$?

This problem was considered by many early psychologists and anatomists, but the most influential suggestion was made by Donald Hebb, who wrote:-

When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency as one of the cells firing B is increased.

In modern terms we would say that the synaptic connection from A to B gets stronger whenever A's firing contributes to B's firing. One could replace "contributes to" by "tends to cause" or even "causes". However, here one must be careful. The synapses between A and B know nothing about the actual causes of B's firing. The only local information that is present at the synapses themselves is the firing of A and the firing of B. If the connection is initially very weak, then A could contribute very little to the firing of B, which must be due to other inputs to B. But from the A-B synapse's point of view, it appears that A causes B's firing whenever the firing of B follows the firing of A after a suitable synaptic delay. We will return to this point later, but for the moment let's just summarise Hebb's Rule as "neurons that fire together, wire together" – i.e. connections strengthen as a result of correlated spiking across them. At the synaptic level, each pair of pre-post spikes that occur with a suitable time delay would lead to a small increase in the connection strength (or, possibly, a raised probability of a larger strength increase).

It is straightforward to incorporate this idea into our idealized neuron models. Since we are using a rate code, it is natural to think of neurons as firing randomly at some mean rate f (in fact neurons in the brain often do have interval spike intervals that are close to exponential). If A is firing at rate f and B is firing at rate g, what is the probability that an A spike will occur "at the same time as" (i.e. within a fixed short interval) a B spike. It is given by the product of the probability that the neurons will fire a spike within the interval, which is proportional to the product fg. This suggests the following mathematical formulation of Hebb's postulate:

$\Delta w_{i,j} = \alpha f_j g_i$

where α is a constant (called a learning constant because it represents how *much* correlated spikes increase w). This looks reasonable: the rule is *local* (the weight change depends only on the activities arriving at the synapses comprising the weight). It is also about as simple as possible. We will look at the consequences of using this rule in the next lecture.