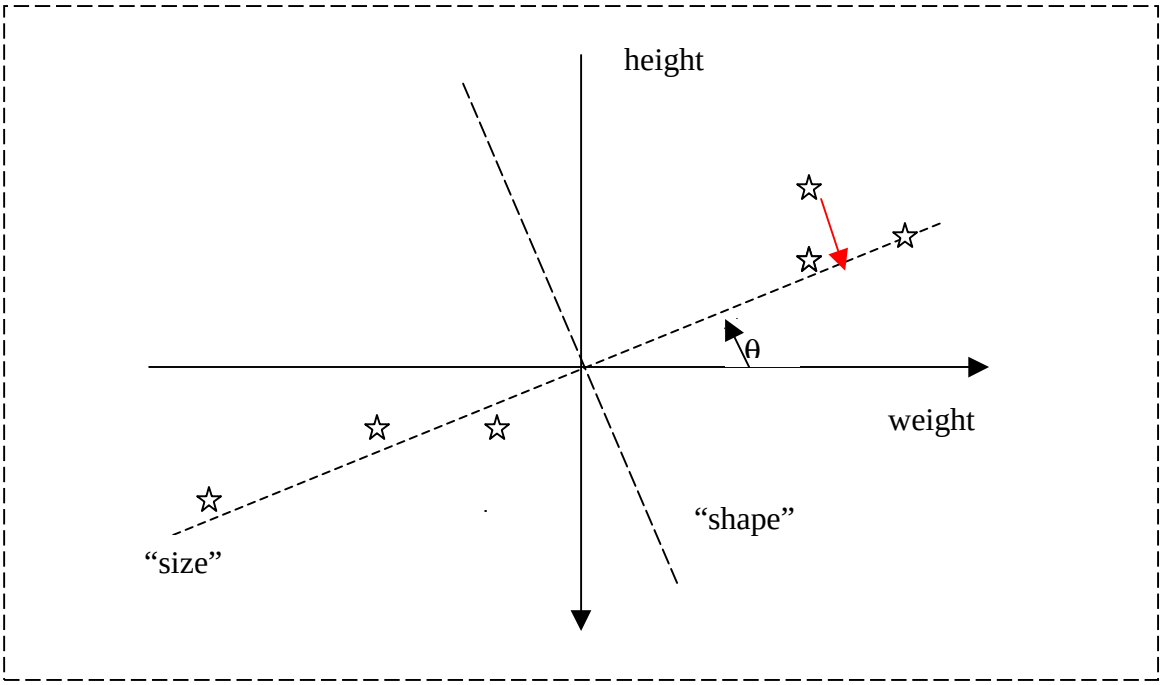
PCA

We have seen that the hippocampus stores patterns of neural activity sent from the neocortex (via the perforant path). What are these neocortical patterns of activity, and how are they generated? One way to answer this question (which we will examine in a later lecture) is to study the physiology and anatomy of the neocortex. This work suggests that the neocortex sets up neural representations of events as they occur, and that these representations facilitate complex computations, resulting in appropriate behavior. However, before looking at the details, we will first explore the general idea that neurons can set up useful representations.

We do not memorise scenes and events like a photograph or a movie. You know what you had for breakfast this morning, but not because your brain took a photograph (which is why we find cameras such useful adjuncts to our lives). You remember that you had bacon and eggs, and perhaps the general look of the eggs on the plate, but not the value of every pixel at every moment as you ate. Your brain represented the complex array of ever-changing retinal pixels in terms of relatively changeless and simple concepts “eggs”, “bacon”, “plate” etc (and perhaps even “easy-over” or “sunny-side up”). In other words, you represented the myriad data points as objects, which, while unique, have recognizable properties. An object, such as an egg or a plate, is essentially a stored prototype, and the brain compares the incoming data (retinal pixels etc) to this prototype, identifying the current collection of pixels as matching one prototype (egg) better than others (plate or chair). The exact way this is done is complex and controversial, but a good starting point for exploration is the statisitical technique known as Principal Component Analysis.

A professor decides to offer a party for his students. Being a neurobiologist, he decides to do things in style, and serve caviar and champagne. But, as a neurobiologist, he cannot afford to provide a huge bowl of caviar for everyone, so he decides to offer each student a portion appropriate to their metabolic requirements. He asks his TA to measure the height and weight of each student, and plots them on a graph (after subtracting the average height and weight). He obtains the following graph



He observes that height and weight (at least for his class) are not unrelated, but correlated, or clustered in the 2 dimensional space defined by height and weight. He fits a straight line relationship (shown as the dashed line), by choosing the line, passing through the origin, for which the sum of the squares of the deviations (one example of a deviation is the length of the arrow shown in red) from the line is minimized – the "least squares fit". He calls this line "size" – a new dimension, which combines the weight and height of a student, in a linear but not necessarily equal way (in the diagram, weight contributes more to size than does height). He then reads off the "size" of each student from the projection of her red arrow on the size line, and gives each student a portion of caviar proportional to her size. The party is a great success.

This is a simple example of Principal Component Analysis, and the "size" line is called the first principal component. The idea is to redescribe a set of measurements using a linear combination of the original set of measurements, in the hope that the new measurements will be simpler and more useful than the original set of measurements. If the old set of measurements are numbers corresponding to weights w and heights h, then the new measurement (in this case size, s) is defined by $s = aw + bh$, where a and b are constants which characterize this particular class of students. The value of s associated with a particular pair of data is simply the projection of the point representing the paired data on the "size" line. The values of a and b were chosen in this case such that the data points scatter evenly on either side of the size line, according to the least squares criterion – this criterion is what defines the s as a PC. However, as we will see, it is possible to use other values of a and b that use some other useful statistical criterion. Since the units of "size" are arbitrary, the "size" is really defined by the angle θ the size line makes with the original measurement axis. For another class of students, for example composed of Sumo wrestlers, a different pair of constants a and b might be needed (giving a different line

and angle). The new measurement, "size", is clearly more economical that the previous pair of measurements, height and weight. Only one set of numbers is needed to characterize the students, rather than 2 sets. Notice however that the new measurement contains less information that the original pair of measurements. This can be seen by trying to reconstruct the original pair of measurements from the new single measurement – all one knows is that the original pair of measurements lies somewhere along the red line, and likely quite close to the size line. In fact the best guess est (w,h) of the original pair (w,h)would be provided simply by "inverting" the size definition, i.e. est w = s/a, est h = s/b. Of course, if the points cluster very tightly along the "size" line, then the reconstruction will be quite accurate. It can be shown that if the scatter in the measured weights and height both follow a Gaussian distribution, then (1) the first PC (in this case "size") provides the best *single* measurement for reconstructing the original data (i.e. a pair of weight and height measurements; of course those paired measurements provide the best possible description of themselves – but just one of those measurements does not allow the other one to be determined) and (2) the average reconstruction error is just the scatter in the points in the direction orthogonal to the first PC (shown as the dashed line marked "shape"), which is called the second PC. Furthermore, if one had both the "size" and "shape" of a student, one could perfectly determine her weight and height. However, the extra precision which the measurement of shape brings is quite small, especially if the scatter in the shape direction is quite small (as already noted). Thus most of the information about a particular student's weight and height is contained in the single size measurement (which is why it was useful for the professor's housekeeping).
PCA is a simple example of "dimension reduction" – encoding high dimension vectors as lower dimension vectors. It is therefore widely used for sending electronic messages – for example sending images over the internet. In the present case the encoding of weight and height as size reduces the dimensions from 2 to 1, with a loss of information that depends on the scatters (i.e. variances) along the original 2 dimensions. If weight and height were uncorrelated (so the points scattered around the height and weight axes), then the single best measurement would still correspond to the first PC, which would be the same as either "weight" or height" (whichever had the highest variance).

PCA is not restricted to 2 dimensional data. The technique was invented by Karl Pearson, who thought that people could be characterized by, and perhaps actually possessed, a general intellectual ability called "g" or "general intelligence". Pearson studied the academic records of large numbers of students, and supplemented them with additional laboratory measurements. Each student therefore corresponds to a particular point in n-dimensional space where n is the number of measurements (e.g math, physics, English, French, geography etc). The whole collection of students define a cloud of points in n dimensional space. Rather than defining a simple "GPA" in which each subject gets equal weight, he chose weights a,b,c etc defining the "least squares" line through the cloud. This is the first PC of the distribution (or "cloud") formed by the student measurements. Of course, this can only be done if there is such a line, i.e. if the points do cluster in n-dimensional space. Pearson found that such a least squares line did exist, and labeled it "general intelligence". Of course this does not really prove that there is a real physical thing called "intelligence".

In this higher dimensional space, there are a large number of directions that are orthogonal to the first PC. Just as the first PC is the directed line upon which the projection of the original data gives the greatest possible variance, the second PC is chosen as the direction, orthogonal to the first PC, for which the projection of the original data has maximal variance. The third, etc PC, down to the nth PC, are constructed in a similar way. The original data can be reconstructed exactly, with no loss of information, from the projections on the entire set of n PCs. However, the greatest amount of information, and hence the best possible reconstruction from a single scalar value, is provided by the first PC, and the lower order PCs provide decreasing amounts of information. Thus the original n dimensional data set can often be compactly described using a limited number of PCs. If the original data has to be transmitted over a relatively few channels (such as an optic nerve or an internet connection), it is efficient to first encode it as a small number of PCs. It can be approximately reconstructed by projecting the reduced and transformed data back to the original coordinates. The JPEG format used to send image files is in principle similar to PCA. However, in the brain there is no need to reconstruct the original data, and the transformed, more compact data can be used in further processing.

More formally PCA can be described in the following way. Suppose we have a random vector **x** whose n components are x_1, x_2 etc (these would be the heights and weights in the first example, and the scores in various subjects in the second example). The elements are assumed to have zero means (i.e. the elements represent the deviation of measurements from their mean values, as in Fig 1). We define the covariance matrix **C** by $\mathbf{C} = \langle \mathbf{x}\,\mathbf{x}^T \rangle$ i.e. the average of the outer product of **x** with itself. The components of **C**, $C_{i,j}$, are approximately the average (over the available instances of **x**, for example the students in examples 1 and 2) of the products $x_i\, x_j$. We define the eigenvectors **e** of C as the solutions of the equation $\mathbf{Cx} = \lambda\mathbf{x}$, i.e. those **x** vectors which when multiplied (from the left) by **C** give vectors that are simply scaled versions of themselves (λ being the scaling factors). These are special vectors that are not rotated by **C**, merely rescaled. Provided that **C** is symmetric (which is guaranteed by its definition, since $C_{i,j} = C_{j,i}$), there will be n different values of λ (λ_1,λ_2 etc, and, correspondingly, n different eigenvectors $\mathbf{e}_1$, $\mathbf{e}_2$ etc**.** The n principal components are the eigenvectors of **C**, with the first PC being the eigenvector for which the eigenvalue λ_1 is largest, etc. The eigenvalues correspond to the variances of the projections of **x** along the eigenvector directions.

Example : the eigenvectors and eigenvalues of the matrix

v c
c v

are 1,1 (eigenvalue c+v) and 1,-1 (eigenvalue v-c). To check this, multiply the matrix by each eigenvector. In each case the 2 elements of the resulting vector are (dot product of first row of matrix with eigenvector) and (dot product of second row of matrix with eigenvector). For the first case, the resulting vector is c+v,v+c which is simply 1,1 scaled by the factor c+v. In the second case the resulting vector is v-c,c-v which is 1,-1 scaled by

the factor v-c. We will use these results to discuss ocular dominance columns in neocortex.

So far we have not discussed the statistical distributions of the elements of **x**, beyond saying that they are random (i.e. picked randomly from some defined distribution). PCA can be done whatever the distribution. However, in the special (but common) case where the elements follow Gaussian distributions, it can be shown that PCA is an optimal way of encoding a vector **x** as a more compact vector (the projections of **x** on the n eigenvectors). First, the projection on the first PC (shown as the red arrow head in Fig 1) is the scalar that best represents the original vector (in several related senses, such as least squares, maximum variance and mutual information – see below) . Second, the projection on the second PC is the scalar that best describes the original vector once the projection on the first PC is known. And so forth for the remaining PCs. In practice, most of the information about the original data can be captured using only a few of the PCs, the remainder can be thrown away, often at great savings of channel capacity, memory etc.

It can be noted that the Central Limit Theorem states that the sum of many independent variables (whatever their distribution) tends to a Gaussian distribution.

It should be noted that PCA is a one-shot technique. It transforms correlated variables (e.g. height and weight) to uncorrelated variables (e.g. size and shape). If PCA is applied again, to the uncorrelated variables, no further change occurs. If the brain only did PCA, it would only need 2 layers of neurons! The lack of effect of further PCA is related to the fact that once an eigenvector has been found, the covariance matrix (which, in the case of Gaussian data, completely characterizes that data) only acts to rescale the transformed variables. Thus PCA is an example of a *decorrelation* technique. It is also often described as a *whitening* technique, based on the following analogy. White light is composed of equal amounts of monochromatic lights of a broad range of visible frequencies. If an object appears white, it is because it reflects equally well all the components of white light. If it appears colored, it reflects more light of one frequency than another eg more green light than blue. Therefore, if an object looks green, the amount of green light it reflects is correlated to the amount of blue light it reflects (i.e. blue light = some fraction of the green light). The analogy is often extended to “white noise”. In the case of a temporally fluctuating signal, white noise sounds like a hiss. White noise is composed of a wide range of frequencies, each of equal power (like white light). If we examine the autocovariance function of white noise (i.e. the average value of a signal multiplied by itself, after imposing a time delay Δt), we find that it is zero (except of course when $\Delta t = 0$). It is uncorrelated. In PCA, instead of looking at a single signal at different times, we are looking at several signals (the PCs) at the same time, but because the signals are uncorrelated, we say that they have been whitened. Whitening is a valuable data preprocessing step, because it ensures that signals are being conveyed efficiently – indeed, for Gaussian signals, optimally.

That PCA is optimal for Gaussian data can be shown in several different ways:

1. Least squares reconstruction criterion. PC allows the best possible reconstruction of the original data, in the least squares sense.
2. PCA maximizes the variance of the transformed variables. Thus in example 1, s has greater variance than either h or w. In fact, the eigenvalues are nothing other than the variances of the projections on the PCs.
3. PCA maximizes the Mutual Information between the original data and the transformed data. This is a theoretical measure of how much knowledge of the transformed data gives us knowledge of the original data.

Information theory is concerned with quantitating information (though it does not really address the significance or utility of information). Its most basic idea is that of *entropy* (represented by H), which is a measure of the uncertainty(or surprise) associated with a signal (or, conversely, of the information that is acquired when learning the exact value of a signal). Suppose the signal is a binary variable that takes a value 1 with probability p and 0 with probability 1-p. (A simple example is tossing a biased coin, where the probability of a head (1) is different from that of a tail (0)). Clearly if p = 1 we know (without examining it) that the signal will always be 1 – its uncertainty is zero (and we gain know information by learning its value). Also, if p = 1, uncertainty is zero. The maximum uncertainty is when p = 0.5 (and this is when we learn the most from looking at the signal), as is the case with an unbiased coin. If the coin has heads on both sides, then p = 1 and there is no point in looking at the tossed coin. If we had 2 binary signals A and B which were completely independent of each other, with "on" probabilities p_1 and p_2, then we would like the information gained by observing that both signals are "on" to be the sum of the information gained by observing each, i.e.

$h_{1+2} = h_1 + h_2$

Since we know that for independent signals the probability that both are "on" is p_1p_2 this suggests that h should be a logarithmic function of p i.e.

$h_{1+2} = - \log p_1p_2 = h_1 + h_2 = - (\log p_1 + \log p_2)$

We define the entropy of a signal to be the average value of the negative log of the probability p_i of observing the signal value i:

$H = \langle -\log p \rangle = -\Sigma\, p_i \log p_i$

(Note that since p <1 log p is negative and H is positive). Given this definition of entropy, we find that using the base 2 for the logarithms, the entropy associated with a binary variable is 0 if p = 0 or 1 (since log 1 = 0), and 1 when p = 0.5 (since $2^{-1} = 0.5$). This is nice, because it means we gain exactly 1 "bit" of information when we learn whether a tossed coin is heads or tails.

Going back to observing a pair of binary variables, what happens if they are *not* independent? Clearly the information associated with learning the value of A would be lessened by learning the value of B (and if A and B were completely correlated, observing A knowing B would be completely uninformative). We write the *conditional* entropy of A knowing B as H(A|B), it is given by log $(p_1|p_2)$ where "$p_1|p_2$" means "the probability that A is 1 given that we know B is 1".

We would like to know how much information about B observing A tells us. This is defined as the information we get by observing A alone minus the information we get by observing A given that we already know B, and it is called the Mutual Information between A and B (symbol I(A,B)). It will be given by the following equation:

I (A,B) = uncertainty A alone – uncertainty A given B = H(A) – H(A|B) = $-\langle \log p_1 \rangle + \langle \log p_1|p_2 \rangle$

Note that we get less information (less negative entropy) by observing A if we already know B (unless A and B are independent). Also, the entropy of observations of B depend on observations of A in exactly the same way i.e. I(A|B) = I(B|A)

Let us now consider the mutual information between the first PC variable s and the original h and w variables. This measures how much "size" tells us about height and weight.

I (s|s,w) = H (s) + H (s|h,s)

Now in this case, the value of s depends completely on h and w (since it is given by the projection of the height-weight vector on the size vector). So the second term is zero. Thus

I (s|s,w) = $\langle \log g(s) \rangle$

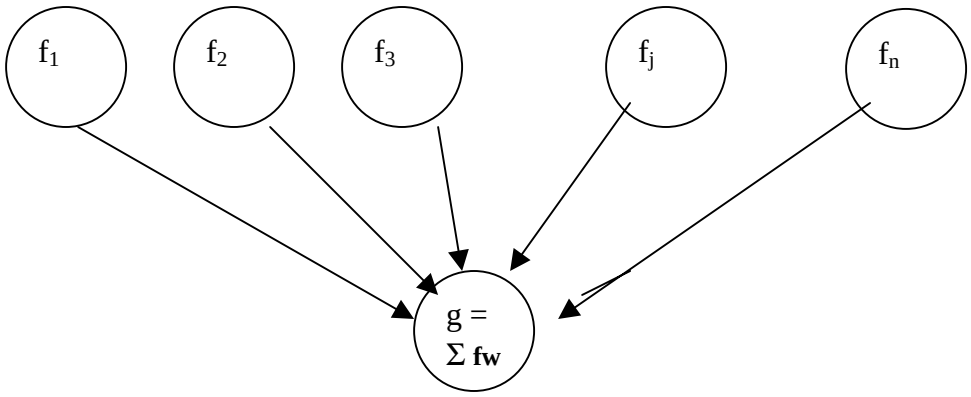
where g is the Gaussian function

$g(s) = (\exp - s^2/2\sigma^2)/(2\pi)^{0.5}\sigma$

Now log (exp x) = x, and the expected value of s^2 is σ^2, so I = $\log \sigma (2\pi)^{0.5}\sigma /2 + (\log_2 e)/2$. Since we know that the first PC has maximal variance, it follows that PC also maximizes the mutual information between the original variables and the transformed variables.

What if the inputs are not Gaussian? In that case the covariance matrix no longer completely characterizes the input statistics, and higher order terms must also be considered (i.e. triple and higher products of input elements). These higher order terms can be captured using a generalization of PCA called Independent Component Analysis.

Can a neural network learn to do PCA? The fact that the covariance matrix is given by the averaged outer product of the input vector elements suggests that it can, since the Hebb rule computes outer products of input vectors. Consider the following simple network (essentially the idealized connectionist neuron we studied in lecture 10):



Idealised connectionist neuron. The neuron gives an output g which is the sum of its weighted inputs f_j.

Consider what happens if we apply a succession of input vectors to the network, each time modifying the synapses using the Hebb rule. This means that each time we show the network a new pattern, the postsynaptic response will be different, and so the weight changes will be different. Even if we showed the network a pattern that it had previously seen, the weight vector update will not be the same, because the weights (and hence the output g) will be different due to learning of all the intervening patterns. Will the network eventually stabilise at some fixed weight vector, or will the weight vector fluctuate indefinitely? If it fluctuates, will it fluctuate around some average vector, or will it change randomly, eventually visiting all possible weights? Obviously this is rather similar to the question of whether a recurrent network will eventually stabilize.

Consider the change in a particular weight (eg w_1) due to a particular pattern eg. $\mathbf{f}^1$. By the Hebb rule this is given by

$$\Delta w_1^{\,1} = k\ g\ f_1^{\,1} = k\ f_1^{\,1}\ (\mathbf{f}^1 \cdot \mathbf{w}) = k\ f_1^{\,1}\ (\mathbf{f}^{1T}\mathbf{w})$$

where **w** is the current weight vector and k is a learning constant. Now **w** reflects the sum of all the previous weight changes, which can be calculated as follows.

The change in the entire weight vector is given by

$\Delta\mathbf{w} = k\,\mathbf{f}^1\,(\mathbf{f}^{T1}.\mathbf{w}) = k\,(\mathbf{f}^1\,\mathbf{f}^{1T}).\mathbf{w} = k\mathbf{M}^1\mathbf{w}$

where $\mathbf{M}^1$ is the symmetric matrix given by the outer product of the current input vector with itself.

We will now consider the average value of the weight vector $\langle\mathbf{w}\rangle$ and its change $\langle\Delta\mathbf{w}\rangle$ given by averaging over all possible input patterns (or at least a large number). We will assume that k is sufficiently small that a statistically large sample of patterns is learned before there are large changes in the weights. We get

$d\langle\mathbf{w}\rangle/dt = k\langle\mathbf{M}\mathbf{w}\rangle = k\langle\mathbf{M}\rangle\langle\mathbf{w}\rangle = k\mathbf{C}\langle\mathbf{w}\rangle$ **…….. (1)**

where we have written $\mathbf{C} = \langle\mathbf{M}\rangle = \langle\mathbf{f}\,\mathbf{f}^T\rangle$. Note that **C** is just the input covariance matrix. This equation implies that the weights will increase indefinitely. This can be seen particularly clearly in the case of a single input neuron, so that Eq 1 reduced to

$d\langle w\rangle/dt = \sigma^2\langle w\rangle$ …….eq 2

where σ^2 is the variance of the input neuron's activity. The average weight will grow exponentially (though in a jerky fashion, depending on the exact sequence of inputs).

An obvious solution in this case would be to modify the rule so that eventually the weight stops growing. This could be achieved by modifying Eq 2 to

$d\langle w\rangle/dt = \sigma^2\langle w\rangle - \sigma^2\langle w_e\rangle$

where $\langle w_e\rangle$ is the average final or equilibrium value of the weight. We therefore optimistically (and, by a deeper analysis, justifiably) modify Eq 1 to

$d\langle\mathbf{w}\rangle/dt = \mathbf{C}\langle\mathbf{w}\rangle - \sigma^2\langle\mathbf{w}_e\rangle$

The original Hebb rule is therefore modified to

$\Delta w_i = k\,g\,f_i - g^2\,w_i$

which is called the Oja rule. It says that the weight change is given by a Hebbian term minus a term that depends on the square of the output activity and on the current weight. This second term is nonHebbian but still local, depending only on quantities that are available at the synapses themselves. One could imagine for example that postsynaptic activity could lead to a nonlinear spine head voltage-dependent Ca increase, which could increase the probability that a synapse would disappear. If the synaptic weight reflected

the number of synapses, this would implement this form of activity- and weight-dependent LTD.

At equilibrium we have $d\langle \mathbf{w} \rangle / dt = 0$, so

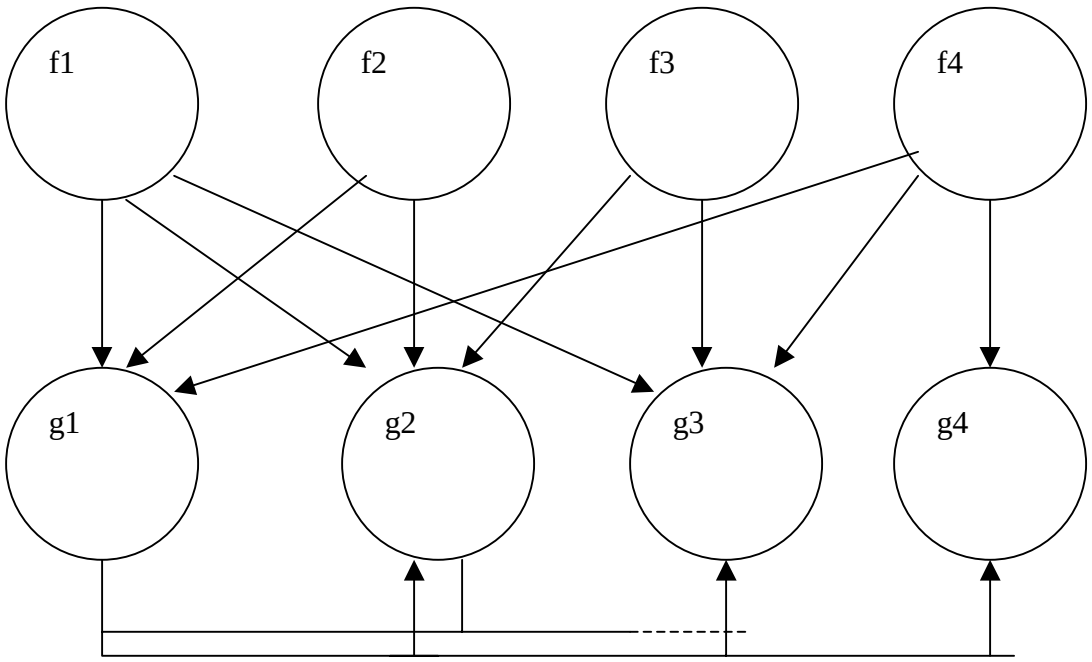
$$\mathbf{C}\langle \mathbf{w}_e \rangle = \sigma^2 \langle \mathbf{w}_e \rangle \quad \text{.....Eq 2}$$

which is identical to the PCA equation $\mathbf{Cx} = \lambda \mathbf{x}$ with $\lambda_1 = \sigma^2$ and $\langle \mathbf{w} \rangle = \mathbf{x}$.

Thus the network we have just analysed finds a Principal Component of the input patterns. A more subtle argument shows that the average weight vector direction is not just a principal component but the first principal component.

What is the meaning of this mumbojumbo? Every time we show a new pattern to the network the weight vector moves in the direction of that pattern, because each weight increases or decreases according to the corresponding element of the pattern. However, if we just used a use-dependent rule (as in the summed vector model) the weight vector would, on average, not change, because we assumed that the input vectors had zero-mean. But using a Hebbian rule means in effect that the extent of the change in the weight vector as it moves toward the current pattern depends on the output of the postsynaptic cell, which in turn measures how similar the current pattern is to the current weight vector. In other words, patterns that most resemble the current weight vector have the largest effect on the weight vector. The weight vector therefore converges to the most representative (or prototypical) pattern, which lies in the direction of the principal component. The output of the neuron is just the projection of a particular input pattern onto the first principal component – in other words it tells us how closely the current pattern resembles the prototype. This output will fluctuate from pattern to pattern, since none of them exactly resemble the prototype. (This incidentally demolishes a common objection to "associationism" – the objection that one can only learn specific instances of things, not idealizations which one has never actually seen). It also illustrates that the Oja rule does "implicit" or "procedural" learning, not "declarative" or "explicit" learning. None of the actual patterns was learned; instead an idealized pattern that was never seen was learned. We can also see what "declarative" memory might involve: not learning of particular sensory patterns (e.g. the pattern of light on the retina) but instead learning the particular outputs of a set of prototype neurons (like the Oja neuron) evoked by a particular sensory pattern.

What about the other principal components? Obviously if we just duplicate this postsynaptic neuron, we would just duplicate the first principal component. There are various ways to force the other n-1 postsynaptic neurons to find the other PCs. For example, one can make asymmetrical recurrent connections from neurons computing higher order PCs to neurons computing lower order PCs:



A network computing the full set of 4 PCs g1, g2 etc from the 4 input neurons f1, f2 etc. The left postsynaptic cell computes the first PC etc. Not all the feedforward and lateral (recurrent) connections are shown

Each higher order neuron inhibits each lower order neuron; also the recurrent weights are learned using an antiHebbian rule, forcing the neurons to be uncorrelated. Thus the second neuron is prevented from learning the first PC, and so learns the second PC, etc.

This extended network can therefore embody all the PCs. The outputs of the neurons are just the projections of the input patterns on the complete set of PCs. These outputs can be used to reconstruct exactly the input pattern, although that is not really a useful thing for the brain to do (except perhaps for witnesses in court cases – it is because the brain is not good at this reconstruction task that witness testimony is so unreliable). Also, most of the useful information is contained in the first few principal components.

It should be noted that because each neuron uses the Oja rule, the weight vectors for each neuron have unit length, and therefore the output variances of each neuron are equal. (This does not pose any problem for reconstruction: the reconstruction procedure uses additional knowledge both about what the eigenvector is and what its eigenvalue is: in reconstructing the heights and weights of our class of students we need to know not just

the projections on the size vector, but also the direction of that vector). The output from such a network is said (because of the variance equalization) to be whitened. This corresponds to the fact that all the outputs are uncorrelated with each other (though they may not be independent of each other). If these outputs are applied to another PC network, there will be no change (because the covariance matrix of these inputs will be zero everywhere but for the uniform diagonal elements). This illustrates the point previously made, that PCA is a one shot deal.