Problem Set: Hebbian Learning

1. **Formulation of constraints.** Start with the equation for the evolution of synaptic weights $\mathbf{w}$ onto one postsynaptic cell: $\tau_w \frac{d}{dt}\mathbf{w} = \mathbf{f}(\mathbf{w})$. We want to constrain the dynamics to live on a constraint surface $\mathbf{c} \cdot \mathbf{w} = k$ for some constraint vector $\mathbf{c}$ and constant k. For example if $\mathbf{c}$ is proportional to the vector of all 1's, the constraint conserves the sum over the weights, $\sum_i w_i$. Note that $\mathbf{c}$ is perpendicular to the constraint surface: along the constraint surface, the value of $\mathbf{c} \cdot \mathbf{w}$ is not changed, meaning that the constraint surface consists locally of the directions perpendicular to $\mathbf{c}$.

   To achieve this constraint, we subtract from $\mathbf{f}(\mathbf{w})$ a vector in the direction $\mathbf{s}$, where $\mathbf{s}$ is some subtracted vector with $\mathbf{s} \cdot \mathbf{c} > 0$. We will take these vectors to have unit length, $|\mathbf{c}| = |\mathbf{s}| = 1$. Note that if $\mathbf{s} = \mathbf{c}$, the constraint is enforced by perpendicular projection onto the constraint surface.

   We assume we start on the constraint surface, so we just have to constrain the derivative vector to point along the constraint surface, that is, to have no $\mathbf{c}$ component: $\mathbf{c} \cdot \frac{d}{dt}\mathbf{w} = 0$.

   (a) Formulate the constrained equation that eliminates the $\mathbf{c}$ component of $\mathbf{f}(\mathbf{w})$ by subtraction of a multiple of $\mathbf{s}$. This will be of the form $\tau_w \frac{d}{dt}\mathbf{w} = \mathbf{f}(\mathbf{w}) - \lambda \mathbf{s}\,(\mathbf{c} \cdot \mathbf{f}(\mathbf{w}))$. What is λ to enforce the constraint? (Hint: see footnote[1].) You might want to also solve this geometrically: consider a two-D space, draw the constraint line that contains $\mathbf{w}$, with its normal vector $\mathbf{c}$; and draw the derivative vector $\mathbf{f}(\mathbf{w})$ arising from $\mathbf{w}$ on the constraint surface, and the vector $\lambda\mathbf{s}$ that moves from the derivative back to reach the constraint surface, and determine the value of λ.

   Note that if $\mathbf{s}$ is a constant this is subtractive normalization (the constraint is enforced by subtracting a fixed amount from each synapse); while if $\mathbf{s} \propto \mathbf{w}$, this is multiplicative normalization (the constraint is enforced by subtracting the weight times a constant, which is equivalent to multiplying the weights by 1 minus that constant).

   (b) Form the projection operator $\mathbf{P} = \mathbf{1} - \frac{\mathbf{s}\mathbf{c}^{\mathrm{T}}}{\mathbf{s}^{\mathrm{T}}\mathbf{c}}$ where $\mathbf{1}$ is the identity matrix. Directly show that this is a projection operator, *i.e.* that $\mathbf{P}^2 = \mathbf{P}$. Also show this as follows: show that $\mathbf{P}$ projects out the $\mathbf{c}$ component of any vector $\mathbf{v}$, *i.e.* $\mathbf{c}^{\mathrm{T}}\mathbf{P}\mathbf{v} = 0$; and that if $\mathbf{c}^{\mathrm{T}}\mathbf{v} = 0$, then $\mathbf{P}\mathbf{v} = \mathbf{v}$. Therefore $\mathbf{P}(\mathbf{P}\mathbf{v}) = \mathbf{P}\mathbf{v}$ for any vector $\mathbf{v}$, hence $\mathbf{P}^2 = \mathbf{P}$. Show that the constrained dynamics can be rewritten $\tau_w \frac{d}{dt}\mathbf{w} = \mathbf{P}\mathbf{f}(\mathbf{w})$.

   (c) Show that the constraint $\mathbf{w} \cdot \mathbf{w} = k$ can be enforced by using $\mathbf{c} = \mathbf{s} = \hat{\mathbf{w}}$, where $\hat{\mathbf{w}} = \mathbf{w}/|\mathbf{w}|$, which is both multiplicative normalization and perpendicular projection onto the constraint surface.

   (d) Let $\mathbf{n}$ be the vector of all 1's, so that $\mathbf{w} \cdot \mathbf{n} = \sum_i w_i$, and let $\hat{\mathbf{n}} = \mathbf{n}/|\mathbf{n}|$. Consider subtractive normalization by perpendicular projection, with $\mathbf{c} = \mathbf{s} = \hat{\mathbf{n}}$. Show that if $\mathbf{w}$

[1] Hint: Dot both sides with $\mathbf{c}$, this must give 0.

is on the constraint surface given by $\mathbf{w} \cdot \hat{\mathbf{n}} = w_n$, then $\mathbf{w} = \mathbf{P}\mathbf{w} + w_n\hat{\mathbf{n}}$. For $\mathbf{f}(\mathbf{w}) = \mathbf{C}\mathbf{w}$, then the equation $\tau_w \frac{d}{dt}\mathbf{w} = \mathbf{P}\mathbf{C}\mathbf{w}$ becomes $\tau_w \frac{d}{dt}\mathbf{w} = \mathbf{P}\mathbf{C}\mathbf{P}\mathbf{w} + w_n\mathbf{P}\mathbf{C}\mathbf{n}$.

2. **Lagrange multipliers.** Suppose we are given an energy function $E(\mathbf{w})$ that we want to minimize subject to some constraint $g(\mathbf{w}) = 0$. As discussed in class, the Lagrange multiplier method says to modify E to $E_c(\mathbf{w}) = E(\mathbf{w}) - \lambda g(\mathbf{w})$. You then minimize $E_c(\mathbf{w})$ with respect to $\mathbf{w}$, by solving $\nabla_{\mathbf{w}} E_c(\mathbf{w}) = 0$, and then choose λ to enforce the constraint. (Note, you only need to include the $\mathbf{w}$-dependent terms in $g(\mathbf{w})$, any other terms are irrelevant to the derivative.)[2]

   Consider weight dynamics that does gradient descent in E, $\nabla_{\mathbf{w}} E = -\frac{d}{dt}\mathbf{w}$. For example, if $E = -\frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{C}\mathbf{w}$ with $\mathbf{C}$ symmetric, this gives $\frac{d}{dt}\mathbf{w} = \mathbf{C}\mathbf{w}$. Then we know that $\frac{d}{dt}E = (\nabla_{\mathbf{w}} E)\left(\frac{d}{dt}\mathbf{w}\right) = -\left(\frac{d}{dt}\mathbf{w}\right)^2 \leq 0$ (strictly < 0 except where $\frac{d}{dt}\mathbf{w} = 0$, *i.e.* at a fixed point), so if E is bounded from below, it is a Lyapunov function.

   Suppose we want to minimize E subject to a constraint $g_1(\mathbf{w}) = \mathbf{w} \cdot \mathbf{n} - k$ or a constraint $g_2(\mathbf{w}) = \mathbf{w} \cdot \mathbf{w} - k$, and we use gradient descent dynamics to minimize $E_c(\mathbf{w})$. Compared to the unconstrained gradient descent dynamics $\nabla_{\mathbf{w}} E = -\frac{d}{dt}\mathbf{w}$, show that gradient descent dynamics on E_c adds a constraint that is enforced by locally perpendicular projection onto the constraint surface. Verify that it gives the constraints that we worked out in the first problem for the following cases: for $g(\mathbf{w}) = \mathbf{w} \cdot \mathbf{n} - k$ it gives the subtractive constraint, and for $g(\mathbf{w}) = \mathbf{w} \cdot \mathbf{w} - k$ it gives the multiplicative constraint on $\mathbf{w} \cdot \mathbf{w}$.

3. **Eigenvectors.** In class we saw that simple Hebbian dynamics can lead to an unconstrained equation $\frac{d}{dt}\mathbf{w} = \mathbf{C}\mathbf{w}$ where C is a symmetric matrix representing some measure of correlations in the inputs. We imagine the inputs are arranged in 2-D space, and that C_{ij} depends only on the spatial distance between inputs i and j. The spatial arrangement of the inputs may be irregular, but we assume that inputs are dense enough that we can convert to a regular spatial grid by averaging the weights in each pixel of our regular grid. We let $\mathbf{r}$ be a two-dimensional position on our grid, $v(\mathbf{r})$ represent the average weight in the pixel associated with $\mathbf{r}$ (so $\mathbf{v}$ is a vector with components $v(\mathbf{r})$, just as $\mathbf{w}$ was a vector with components w_i), and $A(\mathbf{r})$ be the density of such synapses. Thus $A(\mathbf{r})v(\mathbf{r})$ is the total synaptic strength in the given pixel. We let $C(\mathbf{r}, \mathbf{r}')$ represent the average correlation between the inputs at $\mathbf{r}$ and at $\mathbf{r}'$, which we assume depends only on $|\mathbf{r} - \mathbf{r}'|$ so we write this as $C(\mathbf{r} - \mathbf{r}')$, or as $C(\mathbf{d})$ where $\mathbf{d}$ represents the distance between two inputs. Then our equation in terms of the individual weights, $\frac{d}{dt}w_i = \sum_j C_{ij}w_j$, becomes $\frac{d}{dt}v(\mathbf{r}) = \sum_{\mathbf{r}'} C(\mathbf{r} - \mathbf{r}')A(\mathbf{r}')v(\mathbf{r}')$ We can make

[2] Recall from class that the effect of the Lagrange multiplier is to set $\nabla_{\mathbf{w}} E(\mathbf{w}) \propto \nabla_{\mathbf{w}} g(\mathbf{w})$ rather than $= 0$. That is, the LaGrange multiplier says that the derivative of $E(\mathbf{w})$, rather than being 0, can be nonzero but must be locally perpendicular to the constraint surface. Said another way, the component of $\nabla_{\mathbf{w}} E(\mathbf{w})$ along the constraint surface must be 0; $\mathbf{w}$ must be a local minimum of $E(\mathbf{w})$ on the constraint surface, but we don't care if we could make $E(\mathbf{w})$ smaller by moving off the constraint surface. We then choose λ to make sure we're on the correct constraint surface, *i.e.* that $g(\mathbf{w}) = 0$ rather than $g(\mathbf{w}) = q \neq 0$.

this symmetric under interchange of r and r' by changing coordinates to $t(\mathbf{r}) = \sqrt{A(\mathbf{r})}v(\mathbf{r})$, yielding $\frac{d}{dt}t(\mathbf{r}) = \sum_{\mathbf{r}'} \sqrt{A(\mathbf{r})}C(\mathbf{r}-\mathbf{r}')\sqrt{A(\mathbf{r}')}t(\mathbf{r}')$.

Consider a square $N \times N$ grid of inputs, with N an odd number so the center is well-defined (say, $N = 21$), and call the center $\mathbf{r} = 0$. You could either let $A(\mathbf{r}) = A(|\mathbf{r}|)$ be a Gaussian with standard deviation in the range 1/2-1/4 of the grid radius $r_g = (N-1)/2$, or set $A(\mathbf{r})$ to be uniform over a circle of radius r_g and zero outside; doing one of these is sufficient, unless you're curious to try both. Consider two cases for the correlation function: $C(\mathbf{r}) = C(|\mathbf{r}|)$ will be either a Gaussian, or a difference of Gaussians with the negative Gaussian having $2-4$ times wider standard deviation than the positive (and each Gaussian normalized by its variance so the excitatory and inhibitory Gaussians have equal 2D integrals), with standard deviation of the inhibitory Gaussian in the same range as for the arbor function, and standard deviation of the excitatory Gaussian the same in both cases.

Numerically compute the eigenfunctions of the unconstrained operator in each case. It's simplest to use the symmetric representation. You have an effective interaction $\mathbf{L}(\mathbf{r}, \mathbf{r}') = \sqrt{A(\mathbf{r})}C(\mathbf{r}-\mathbf{r}')\sqrt{A(\mathbf{r}')}$. Because space is two-dimensional, $\mathbf{t}$ is a matrix and $\mathbf{L}$ has four indices. You want to unfold $\mathbf{t}$ to be a one-dimensional array of synapses, correspondingly compute a matrix version of $\mathbf{L}$ which is the interaction between each pair of synapses (a symmetric matrix), and compute the eigenvectors of this matrix. You then want to fold the eigenvectors back up to see what they look like in two dimensions, and convert them back to the $\mathbf{v}$ representation.

Plot the eigenvectors corresponding to the top ten or so eigenvalues, and note their eigenvalues. They should look like a product of a radial function and an angular function. Verify that the modes (I use 'mode' as another word for 'eigenvector') with angular nodes have approximately zero summed weight, where summed weight is the product of the number of synapses at a point times their average value, or $\sum_{\mathbf{r}} A(\mathbf{r})e_v(\mathbf{r}) = \sum_{\mathbf{r}} \sqrt{A(\mathbf{r})}e_t(\mathbf{r})$ where $\mathbf{e}_v$ and $\mathbf{e}_t$ are the eigenvector in the $\mathbf{v}$ or $\mathbf{t}$ basis respectively.[3] (The summed weight would be exactly zero if our interaction $\mathbf{L}(\mathbf{r}, \mathbf{r}')$ were circularly symmetric; but it will only approximately sum to zero, because the circular symmetry is only approximate on the square grid.) Verify that the modes without angular nodes typically have non-zero summed weight when C is a single Gaussian; what about when C is a difference of Gaussians? For the case that $\mathbf{C}$ is a single Gaussian, verify that the eigenvalues are ordered so that adding a node always lowers the eigenvalue. (Optional: For the case that C is a difference of Gaussians, verify that the spatial periodicity of the leading eigenvectors is roughly equal to the period corresponding to the peak of the Fourier transform of $C(\mathbf{r})$. If you're not comfortable with Fourier transforms, you could just note that the spatial period of the receptive field is similar to that of $C(\mathbf{r})$.)

[3]Note: the eigenvector $\mathbf{e}_v$ is a product of a radial and an angular function and $A(\mathbf{r})$ only depends on the radial variable, so multiplying the eigenvector $e_v(\mathbf{r})$ by $A(\mathbf{r})$ only modifies the radial function. The angular function with at least one angular node ensures zero sum at each radial position, so the multiplication by $A(\mathbf{r})$ actually does not alter whether modes with angular nodes are zero-sum.

Show that the sum of (i) the mode with $n > 0$ radial nodes and no angular nodes and (ii) the mode with $n + 1$ radial nodes and two angular nodes gives an oriented, simple-cell-like receptive field. (You can show this for a single choice of $n > 0$.) These two modes should be roughly degenerate (have roughly the same eigenvalue); if they were exactly degenerate, then their sum would also be an eigenvector with the same eigenvalue, and so we could consider the simple-cell-like receptive field to be an eigenvector.

Now compute the eigenvectors for the subtractively constrained version of the operator. We constrain the total weight, $\sum_{\mathbf{r}} A(\mathbf{r})\mathbf{v}(\mathbf{r})$ or $\sum_{\mathbf{r}} \sqrt{A(\mathbf{r})}\mathbf{t}(\mathbf{r})$, so in the $\mathbf{t}$ representation the constraint vector is $\mathbf{c} \propto \sqrt{\mathbf{A}}$ where $\mathbf{A}$ is the vector with elements $A(\mathbf{r})$ and the squareroot is understood to be applied element-by-element to the vector. Verify that $\sum_{\mathbf{r}} \sqrt{A(\mathbf{r})}\mathbf{t}(\mathbf{r})$ is conserved by the equation

$$\frac{d}{dt}\mathbf{t}(\mathbf{r}) = \sqrt{A(\mathbf{r})} \sum_{\mathbf{r}'} \left[C(\mathbf{r} - \mathbf{r}')\sqrt{A(\mathbf{r}')}) - \frac{\sum_{\mathbf{r}''} A(\mathbf{r}'')C(\mathbf{r}'' - \mathbf{r}')\sqrt{A(\mathbf{r}')}}{\sum_{\mathbf{s}} A(\mathbf{s})} \right] \mathbf{t}(\mathbf{r}') \qquad (1)$$

i.e. that under this equation $\frac{d}{dt}\sum_{\mathbf{r}} \sqrt{A(\mathbf{r})}\mathbf{t}(\mathbf{r}) = 0$. Show that the operator multiplying $\mathbf{t}(r')$ on the right is $\mathbf{PL}$ where $\mathbf{L}$ is defined above and $\mathbf{P} = \mathbf{1} - \frac{\sqrt{\mathbf{A}}\sqrt{\mathbf{A}}^T}{\sqrt{\mathbf{A}}^T\sqrt{\mathbf{A}}}$.

Which modes are modified by the constraint? How are they and their eigenvalues modified? Are all modes now zero-sum?

You should find: the modes with angular nodes, which were zero-sum without the constraint, are still eigenvectors of the constrained operator. This is because they pass through the constraint unchanged, so if they are eigenvectors of $\mathbf{L}$, they are eigenvectors of $\mathbf{PLP}$. As for the modes without angular nodes: you should find that this mode with a given number of radial nodes is converted to another mode with the same number of radial nodes and the same eigenvalue but is now zero-sum; except that the mode with no nodes, angular or radial, which had the largest eigenvalue in the unconstrained case, is not zero-sum but now has eigenvalue 0 (and may be modified by the constraint). The modes without angular nodes must span the same subspace with the constraint as they did without the constraint, because all the other modes are unchanged; but they do so with a set of zero-sum vectors, plus one non-zero-sum vector with eigenvalue zero.

If you want to venture into this in more detail, it is analyzed in a 1990 paper by myself and David MacKay, which I will put on the web site; and a lot of related material on the role of constraints is covered in a subsequent 1994 paper that we wrote, that I will also put there.

4. **Optional: Simulate the dynamics.** This is just if you want to explore further, and see how the outcome of the dynamics relates to the eigenvectors you have computed.

   You will want to start with a random initial condition, say weights $v(\mathbf{r})$ uniformly distributed between 0.8 and 1.2.

   Things to potentially explore:

- If you multiplicatively constrain (at each timestep, after modifying the weights according to the unconstrained equation, multiply all synapses by a common factor to restore the weight sum to the constrained level), your final weight vector will be proportional to the principal eigenvector. If you don't allow weights to go negative (stop negative changes in weights at 0, before multiplicatively renormalizing), this will modify the result for the difference-of-Gaussians C, but not for the Gaussian C.
- If you simulate the subtractively constrained equation: if you have a lower weight limit at zero, weights will evolve until only one weight is nonzero. So you probably want to have an upper limit, say at 4, as well as a lower limit at 0. In this case all, or all but one, synapse will go to either the upper or the lower limit. However you run into a problem in this case: if you update weights according to the subtractively constrained dynamics, and then set all weights > 4 to 4 and all weights < 0 to 0, you will have changed the weight sum. To compensate for this, you could do a last multiplicative renormalization, multiplying all the weights that are not at upper or lower limits by a common factor to restore the weight sum (and maybe cut off at 4 any that are pushed above 4; you could iterate but maybe stop there). It starts to get klugy but it doesn't qualitatively change the results.
- Under the subtractively constrained dynamics, you will likely find that the final result looks something like the principal eigenvector of the constrained operator, squashed up against the weight limits. However, there is one caveat. For ease of notation let me write $\mathbf{A}_{\surd} = \sqrt{\mathbf{A}}$ and $\hat{\mathbf{A}}_{\surd} = \mathbf{A}_{\surd}/|\mathbf{A}_{\surd}|$. Recall that $\mathbf{w} = \mathbf{P}\mathbf{w} + w_A\hat{\mathbf{A}}_{\surd}$ where $w_A = \mathbf{w} \cdot \hat{\mathbf{A}}_{\surd}$. Thus the dynamics can be written $\frac{d}{dt}\mathbf{w} = \mathbf{PLP}\mathbf{w} + w_A\hat{\mathbf{A}}_{\surd}$. The eigenvectors and eigenvalues of $\mathbf{PL}$ and $\mathbf{PLP}$ with nonzero eigenvalues are identical: if $\mathbf{PL}\mathbf{e}_i = \lambda_i\mathbf{e}_i$, then, multiplying both sides by $\mathbf{P}$ and recalling $\mathbf{P}^2 = \mathbf{P}$, the left side is unchanged, so the right side is unchanged, so if $\lambda_i \neq 0$, then $\mathbf{P}\mathbf{e}_i = \mathbf{e}_i$ (meaning that $\mathbf{e}_i$ is zero sum), and therefore $\mathbf{PLP}\mathbf{e}_i = \mathbf{PL}\mathbf{e}_i = \lambda_i\mathbf{e}_i$. It's a more complicated argument but $\mathbf{PL}$ and $\mathbf{PLP}$ both have one eigenvector with zero eigenvalue and in both cases it is non-zero-sum with no angular or radial nodes. So for practical purposes the eigenvectors and eigenvalues of $\mathbf{PLP}$ and $\mathbf{PL}$ are identical. But the remaining term, $w_n\hat{\mathbf{A}}_{\surd}$, has no angular nodes and so it is a linear combination of the eigenvectors of $\mathbf{PLP}$ with no angular nodes. As a constant contributor to $\frac{d}{dt}\mathbf{w}$ it gives a boost to the growth of those modes. That boost becomes less and less relevant as the leading modes get bigger and bigger, since at each time point they contribute something to $\frac{d}{dt}\mathbf{w}$ proportional to their amplitude, whereas the boosting term does not change over time. So if the weight limits are far enough away from the initial condition that the leading modes have room to grow, then the boosting term will have little effect on the dynamics and the principal constrained eigenvector should dominate. However, if there is less room, one of the modes without angular nodes might get boosted into the winning position even if it is not the principal eigenvector.