MCN 2025, Problem Set 2

1 Maximum *a posteriori* estimation (compressed sensing)

So far, we (implicitly or explicitly) assumed a uniform prior over the quantities being estimated and for the most part used maximum likelihood estimation. Here we will explore the use of priors to regularize our estimation problem. In particular, we will consider the special case where the quantity being estimated is sparse (i.e. with few non-zero elements). L_1 regularization but not L_2 regularization sometimes sets the estimated signal exactly to zero. This exercise develops an intuition for this form of regularizer/prior.

1.1 L_2 regularization in one dimension

You want to estimate a single scalar quantity s from a noisy measurement $x = s + \epsilon$ where ϵ is a zero mean Gaussian noise variable with variance σ^2. Suppose you have a prior on s that is a zero mean, unit variance Gaussian. Then your maximum a-posteriori (MAP) estimate $\hat{s}$ of the unknown signal s is $\hat{s} = \text{argmax}_s p(s|x)$. Show that $\hat{s}$ arises as the solution to the following optimization problem:

$$\hat{s}(x, \sigma^2) = \text{argmin}_s \left(\frac{1}{2} \frac{(s-x)^2}{\sigma^2} + \frac{1}{2} s^2 \right).$$

1.2 Solve the above optimization problem and write down the function $\hat{s}(x, \sigma^2)$.

1.3 L_1 regularization in one dimension

Now let's consider a new prior in which s has a significant probability of being 0 (this is one definition of sparsity). You should see that L_1 regularization yields a *nonlinear* estimate (as a function of x) that is exactly 0 as long as your measurement x is below a noise-dependent threshold. On the other hand, L_2 regularization yields an estimate that is a *linear* function of the measurement x. Replace the L_2 regularization above with the L_1 regularization below to get a different estimate:

$$\hat{s}(x\,, \sigma^2) = \text{argmin}_s \left(\frac{1}{2} \frac{(s-x)^2}{\sigma^2} + \frac{2|s|}{\sqrt{2}} \right).$$

(i) Show that the above equation is the Maximum a posteriori solution $\hat{s} = \text{argmax}_s p(s|x)$ for a Laplace prior

$$p(s) = \frac{1}{\sqrt{2}} e^{-\frac{2|s|}{\sqrt{2}}}.$$

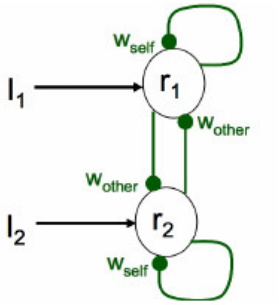
(ii) Prove that

$$\hat{s}(x\,, \sigma^2) = \text{sgn}(x) \left(|x| - \frac{2\sigma^2}{\sqrt{2}} \right)_+,$$

where $(y)_+ = y$ if $y > 0$ and 0 if $y \leq 0$. Also plot $\hat{s}$ as a function of x to see what it looks like.

2 Symmetric Linear Network

Many neural networks, including the oculomotor neural integrator, consist of 2 populations of neurons whose effective connectivity is such that they provide self-excitation to neurons in their own population and inhibit neurons in the opposite population. A simple case of these circuits occurs when the connections are symmetric as in the following diagram:



Here, w_{self} gives the connection between either neuron population and itself and w_{other} gives the connection between the different neuron populations. In this problem, we will quite generally find the eigenvalues and eigenvectors for a circuit with this connectivity, and then apply our findings to several relevant cases.

a. Write down the equations describing this network, assuming the 2 neurons each decay to zero with time constant τ in the absence of input. Write them both as separate equations for dr_1/dt and dr_2/dt and as a single equation written compactly in terms of the firing rate vector $\mathbf{r} = [r_1, r_2]$ and a recurrent connectivity matrix $\mathbf{W}$.

b. Recalling that the eigenvectors of $\mathbf{W}$ are the patterns of inputs with the property that the outputs of the network will be a scaled version of this same pattern, and noting the symmetry of the network, try to *guess* the eigenvectors $\xi^{(1)}$ and $\xi^{(2)}$ of $\mathbf{W}$. (Use the convention that the first element of each eigenvector equals +1.) Confirm that you have found the correct direction by showing that $\mathbf{W}\xi = \lambda\xi$ for each eigenvector and give a formula for the corresponding eigenvalues λ_1 and λ_2.

c. Decompose the input vector $\mathbf{I}$ into a combination of the two eigenvectors $\mathbf{I} = b_1\xi^{(1)} + b_2\xi^{(2)}$ you found in part (b). Writing out this equation for each component, you should find that the coefficients b_1 and b_2, respectively, correspond to a part of the input that is common to both cells, I_{common}, and a part I_{diff} that gives the difference of each cell's input from this common value, i.e. that:

$$I_1 = I_{\text{common}} + I_{\text{diff}}$$
$$I_2 = I_{\text{common}} - I_{\text{diff}}$$

This implies that the common (or equivalently average) and difference portions of the input behave independently.

d. Suppose that each neuron excites itself ($w_{\text{self}} > 0$) and inhibits the other ($w_{\text{other}} < 0$) and that the inhibitory connections are stronger than the self-excitatory connections (i.e. $|w_{\text{other}}| > w_{\text{self}}$).

(1) What will happen to inputs that are common to the two cells? (Will they be amplified or attenuated?) Show this by looking at the eigenvalue for the appropriate eigenvector.

(2) What will happen to inputs that are opposite for the two cells?

(3) Calculate the eigenvalues for $w_{\text{self}} = 0.2$ and $w_{\text{other}} = -0.7$.

(4) Calculate the amplification factors $1/(1-\lambda)$ for each eigenvector, using the values in part (3) above.

(5) If each cell in the network has an intrinsic time constant $\tau = 18$ ms, what will be the corresponding time constants τ_{eff} for each component? From this answer, do you expect the amplified component to change more or less quickly than the attenuated component?

e. Consider a network with $w_{\text{self}} = 0.2$ and $w_{\text{other}} = -0.7$, as in part (d). Suppose $I_1 = 63$ Hz and $I_2 = 57$ Hz.

(1) Write these inputs in the form $\mathbf{I} = b_1\xi^{(1)} + b_2\xi^{(2)}$. What are b_1 and b_2?

(2) Recall that the steady-state firing rate is obtained by simply scaling each component (b_1 or b_2) of $\mathbf{I}$ by the corresponding amplification factor ($(1-\lambda_1)^{-1}$ or $(1-\lambda_2)^{-1}$, respectively), and then adding together these components, i.e.

$$\mathbf{r} = \frac{b_1}{1-\lambda_1}\xi^{(1)} + \frac{b_2}{1-\lambda_2}\xi^{(2)}$$

What will the steady-state firing rate vector $\mathbf{r}_\infty = [r_{\infty,1}, r_{\infty,2}]$ equal? For various initial conditions, sketch the expected trajectory of the firing rates as they approach this steady-state value.

f. Simulate the network by numerically solving the equations; the point is to "blindly" simulate these equations and confirm that you get the same answer you found analytically. For initial conditions, set $\mathbf{r}(t=0) = \mathbf{I}$, i.e. start out the network at the firing rate values that would have been obtained in steady-state if there were no recurrent connections.

Plot r_1 and r_2 as a function of time for long enough to see them reach steady state. On the same set of axes, plot the eigenvectors $\xi^{(1)}$ and $\xi^{(2)}$. Check that the steady state values that you get are the same as those you calculated analytically.

g. **Winner-take-all networks.** For a more extreme example of the rivalry between inputs observed in part (f), see what happens if w_{other} is a large negative value. For this network, add constraints on your neurons such that firing rates that become negative get thresholded to zero. Experiment with various initial conditions and input values.

(i) Such a network is called a *winner-take-all network* because the loser becomes zero and the winner achieves a large value determined by its external input $\mathbf{I}$ and by w_{self}. What is the condition for one of the neurons to be driven to zero firing rate? What additional condition (on the synaptic weights) will lead the winner to exhibit runaway growth? (If you wish to explore this part of the model's parameter space, you can prevent runaway growth from occurring by adding saturation to the model. This is most simply done by imposing a maximum firing rate constraint such that neurons do not increase their firing rates beyond a fixed level $r_{\max}$).

(ii) This model is commonly used as a simple description of phenomena such as perceptual rivalry. A fundamental property of perceptual rivalry is that the percept remains stable for a while and then stochastically switches to the other possibility. Can you find a way to modify the previous model to reproduce this behavior?

3 The Bienenstock Cooper Munro (BCM) Rule

This question is due to Claudia Clopath. Implement a simulation of the competition between two input patterns $x_1 = (20, 0)$ and $x_2 = (0, 20)$. At each timestep, one of the two patterns is presented to the neuron,

the pattern is chosen randomly with a probability 0.5 of being pattern x_1 and 0.5 of being pattern x_2. The weights follow the BCM rule. The output of the neuron at each timestep is given by

$$y(t) = \mathbf{w}^T \mathbf{x}(t)$$

where $\mathbf{x}(t)$ is the input presented at time t.

Recall from Claudia Clopath's lecture that the weight update for the BCM rule is:

$$\frac{d}{dt}\mathbf{w} = \eta\, \mathbf{x}\, y\, (y - \theta)$$

where θ is the sliding threshold: $\theta = \frac{\langle y^2 \rangle}{y_0}$. Note that the average $\langle y \rangle$ must be computed online. The threshold will therefore obey the rule:

$$\tau \frac{d}{dt}\theta = -\theta + \frac{y(t)^2}{y_0}$$

3.1 Implement this simulation using a total time $T = 10\,\text{s}$, $\eta = 1 \times 10^{-6}\,\text{ms}^{-1}$, $y_0 = 10$, $\tau = 50\,\text{ms}$. Use the Euler method with a timestep of 1 ms. You should also put a hard bound for the weights at 0 (when $w_i < 0$, set it to 0).

3.2 Plot the weights $\mathbf{w}$, the sliding threshold θ and the output y. What do you expect to see?

3.3 Run the code a few more times. What do you see? Zoom on the first 500 time steps, what do you see?

3.4 What are each of the parameters η, y_0, and τ? How do the dynamics of plasticity change when each of these parameters is changed? Check your intuition with the simulation.

3.5 Compare and contrast this learning rule with Hebbian plasticity. What advantages does it have? Does it have any disadvantages? **Extra credit:** Simulate the same set up with more traditional Hebbian plasticity, $\frac{d}{dt}\mathbf{w} = \eta \mathbf{x} y$. What happens?