Lecture 21.2 Exercises

The scripts are subdivided into cells using the %% comment sign. Cells can be run separately by clicking them and hitting Ctrl-Enter. Note: these scripts use the function fig to create figures and set default text sizes etc.

21.2.1 Covariance Learning

SpectralBasis
Here the input on the parallel fibres consists of a 1 sec sample of the first nf = 50 harmonics of a sine wave with frequency f = 1/2. Some of these harmonics are plotted in Figure 21.4.

The output to be reconstructed by the Purkinje cell is a 1sec castle shaped response, plotted in Figure 2 of the text.

TimingBasis
Here the inputs are the chaotic outputs of a recurrent neural net. This network serves as a reservoir (see the literature on reservoir computing) supplying a complex basis capable of reconstructing many kinds of behaviour.

Both the scripts implement covariance learning in batch mode

$$\delta w_i = -\beta \left\langle e(t)\, p_i(t) \right\rangle \approx -\beta \frac{1}{N} \sum e(t)\, p_i(t)$$

where the sum is taken over all time steps in the 1sec sample. Since all the PF inputs are stored in the columns of the nt x nf matrix p the sum can be calculated in vectorised form as p'*e

It plots the current output at every step so that the progress of learning can be visualised. You can comment this section out for efficiency.

Optimal Reconstruction
Since this is a linear approximation problem we have access to the optimal solution which we can use to assess the success of the learning algorithm. The next section of code calculates the optimal reconstruction as the least squares solution of the system of equations

P*w=zd

In MatLab this is best obtained using the backslash operator

w = P\zd

1. Change nf to show that larger bases give more accurate reconstructions. Note the problem in accurately fitting the discontinuities even for very large nf. If you are interested in this ask your instructor about Gibb's phenomenon.

2. Change the desired output to something more biologically plausible (e.g. learn to produce a Gaussian response with a specific width and delay).

21.2.2 Stochastic Learning

NoiseCancel
The code inputs a sample of music and contaminates with a sinusoidal signal. In a real problem this signal would have known frequency, but unknown amplitude and phase.

Since we know the contaminating frequency we can cancel the noise using a combination of sine and cosine waves in the correct combination, hence the input on the parallel fibres consists of these two signals.

Stochastic Learning
The script implements covariance learning in continuous (stochastic) mode using only one pass through the data

$$\delta w_i = -\beta \left\langle e(t)\, p_i(t) \right\rangle \approx -\beta \frac{1}{N} e(t)\, p_i(t)$$

It plots the current output every so often during learning so that the progress of learning can be visualised. You can comment this section out for efficiency. The learned filter is applied to the data on a second pass to assess the quality of learning.

1. Try other noise sources, for example you could record your own voice and contaminate the music signal with an unknown amount of this signal.

2. In the simulation the learning rate is set to be very low so that you can visualise the progress of learning. What is the fastest stable learning rate that you can use?

3. Examine what happens if you use a slightly incorrect frequency for the predictor signals (don't be too ambitious at first, try, say, f =440.5 rather than the correct 440Hz) . You should find that for a fast enough learning rate the weights learn quickly enough to track the changing phase difference due to the incorrect frequency and give reasonable noise cancelation. This ability to track a changing world is one of the advantages of an adaptive filter.