Quiz 2, Computational Neuroscience

Handed out: 03/22/2004
Due: 03/30/2004

Rules of the Game

You may take as long as you like to do the problems, but all work, including earlier unfinished work, must be handed in by **5:00 pm on Tuesday, March 30, 2004**. There will be no extensions.

**IMPORTANT:** You may use any resources you wish in order to do the problems. But unlike previous homeworks, **collaboration is NOT allowed**. Do not discuss the problems with any other person, except clarify possible mistakes the instructor made in the quiz questions. To be perfectly clear, collaboration on the quiz is considered cheating and thus can result in failing the course.

As usual, answer each question marked by a bullet within any particular problem.

2.1 Linear Filtering: Edge Enhancement in Images

Use the provided MATLAB package to enhance the edges in the image called `paolina`. To access the image, load the MATLAB data file called `paolina.mat` by typing `load paolina.mat -ascii`. You can display this and other images by using the function `DISPRAW` which we have provided, e.g., by typing `disprawl(paolina)`. All the data and code files should be downloaded from the class website. The `paolina` is a $480 \times 480$ grayscale image. The `paolina.mat` is in simple ASCII format to allow cross platform MATLAB access: it has 480 lines and each line has 480 numbers — of course, you can also write program in other language to read this file and to do the linear filtering, but you need to present you results graphically. If you are successful to this point, you should see an image as the following:

![Image](image_url)

You will use MATLAB to try out some of the edge enhancements techniques on real images. The main thing you will need to do here is designing two dimensional kernels (matrices) to convolve with images. The key to success is to realize that even very coarse approximations to a Gaussian are pretty good, so for example the matrix $[0.368 \ 0.606 \ 0.368 \ ; \ 0.606 \ 1 \ 0.606 \ ; \ 0.368 \ 0.606 \ 0.368]$ is a fine 3x3 Gaussian estimate to use for smoothing. To convince yourself of this, do the following bullet but don’t hand in anything for it (we trust you):
• Show that if you convolve \([1 \ 2 \ 1]\) (or almost any other short symmetric vector) with itself over and over again, the vector you tend towards has an exactly Gaussian profile and that the same is true in two dimensions for say \([1 \ 2 \ 1; \ 2 \ 4 \ 2; \ 1 \ 2 \ 1]\) (or almost any other square symmetric matrix). Muse to yourself about how this is related to the central limit theorem which you should know from probability theory.

• By using the MATLAB commands help and type and by trying out simple examples, briefly explain how the MATLAB functions GRADIENT and DEL2 do their estimation of the gradients and Laplacians of images.

• Use the functions CONV2, GRADIENT, DEL2 to do the following to the supplied images: (1) take the Laplacian with and without smoothing beforehand, (2) take the magnitude of the gradient with and without smoothing beforehand. Hand in one plot of a nicely edge-enhanced version of paolina obtained by using your favourite smoothing kernel and either method (1) or (2) above. Do again all of operation using CONV2 instead of CONV1. You can find the routine in the Data and codes 2 directory in the problem set 2. Then explain what that routine does.

2.2 More on Reconstruction

Use the MATLAB package provided to generate a Gaussian noise stimulus with bandwidth \(B_s\) and standard deviations \(\sigma_s\) using gensig. It should be reasonably long, say, at least, 10,000 data points.

The stimulus generated above is now used as the input to the I & F model of the neuron with \(R = 0.8\), \(C = 0.05\), \(T_{ref} = 0.005\) (where \(R\), \(C\), \(T_{ref}\) are the membrane capacitance, resistance, refractory period, respectively). So generate the output of the integrate-and-fire neuron for the input (using your iaism).

• Use reconst to get the best linear filter for each pair of \(\sigma_s\) and \(B_s\) values (\(\sigma_s = 2,3,4,5,6\) and \(B_s = 25,50,75,100,125Hz\)), and record the mean squared error \(E\) in each case. On one graph, plot \(E\) as a function of stimulus variance \((\sigma_s)\) for different values of stimulus bandwidth \((B_s)\). On another graph, plot \(E\) as a function of stimulus bandwidth \((B_s)\) for different values of stimulus variance \((\sigma_s)\). Briefly explain the behavior of your plot.

• Choose a particular value of \(\sigma_s\) and vary \(B_s\). Obtain the optimal filters in each case. Plot these filters (by this we mean plot the kernels that you get) and see how the properties of the filter vary with \(B_s\). Explain briefly what you see.

• Choose a particular value of \(B_s\) and vary \(\sigma_s\). Obtain the optimal filters in each case. Plot these filters and see how the properties of the filter vary with \(\sigma_s\). Explain briefly what you see.

2.3 Optimal Coding in Early Vision Multi-Dimensional Solution

Things get much more interesting if \(x\) is not a scalar, but an \(N\)-dimensional vector. As an example, \(x\) might be the vector representing light intensity at each of 1,000,000 photoreceptors. Some of the retina's output cells (retinal ganglion cells) seem to pool inputs from several photoreceptors in a roughly linear fashion. In addition to that, the pooling could also include integration over time, say, at LGN level (a stage between retina and visual cortex) — thus spatiotemporal coding. This pooling would be represented by \(A\), our linear channel; the retinal ganglion cells' (or LGN) outputs would be \(y\). A re-phrasing of the basic question, then, would be "given a probability distribution on \(x\), noise models \(n\) and \(u\), and a scaling constraint on \(y\), what is the pooling strategy that would result in the retinal ganglion cells' (or LGN) output being most informative about the light signals \(x\) coming into the eye?"
As we will show later (in a two-dimensional case) that if we look at the Fourier transform of the coding, we get the same solution, except that $|A|, |\sigma^2|$ are now functions of spatiotemporal frequencies. Specifically, $[\sigma^2(f, w)]$ is now the power spectrum of natural time-varying images, or the Fourier transform of the covariance matrix of natural time-varying images. $|A(f, w)|$ is a cell's response to sine wave pattern of spatial frequency $f$ and modulated at temporal frequency $w$.

2.3.1 The Two-Dimensional Case

Here we will explore the relatively simple case $N=2$ with $x$, $n$ and $u$ being gaussian distributions. First we set the stage by defining our terms. We will change variables into a more convenient representation shortly, so we start with upper-case variables. The ones we will mostly work with will be in lower-case.\(^1\)

We have a two-dimensional input vector $X = (X_1, X_2)^T$. We will assume that the inputs are randomly distributed with a 2-dimensional gaussian probability density:

$$p(X) = \frac{1}{2\pi |S|} e^{-\frac{1}{2} X^T S^{-1} X}$$

with $S$, the covariance matrix ($S_{ij} = \langle X_i X_j \rangle$), being

$$S = \begin{pmatrix} \Sigma^2 & \Gamma^2 \\ \Gamma^2 & \Sigma^2 \end{pmatrix}$$

and $|S|$ is just the determinant of $S$. If $X_1$ and $X_2$ are strongly correlated, $\Gamma$ will be almost as large as $\Sigma$; if they are weakly correlated, $\Gamma$ will be small.

Let $N = (N_1, N_2)^T$ be a vector composed of independent gaussian variables\(^2\) each of mean zero and variance $\eta^2$. Let $V = (V_1, V_2)^T$ be a similar vector, but with components with variance $\nu^2$. Let $M$ be the matrix

$$M = \begin{pmatrix} A & B \\ B & A \end{pmatrix}$$

Define the output $Y = (Y_1, Y_2)^T$ to be

$$Y = M(X + N) + V$$

This should look familiar, and is just the general way we pose the problem. We want to find the $M$ which leads to maximum information in $Y$ about $X$. Now define the following transformation to the vectors $y$, $x$, $n$ and $v$, whose components are:

$$\begin{align*}
  y_1 &= \frac{1}{\sqrt{2}}(Y_1 + Y_2) \\
  y_2 &= \frac{1}{\sqrt{2}}(Y_1 - Y_2) \\
  x_1 &= \frac{1}{\sqrt{2}}(X_1 + X_2) \\
  x_2 &= \frac{1}{\sqrt{2}}(X_1 - X_2) \\
  n_1 &= \frac{1}{\sqrt{2}}(N_1 + N_2) \\
  n_2 &= \frac{1}{\sqrt{2}}(N_1 - N_2) \\
  v_1 &= \frac{1}{\sqrt{2}}(V_1 + V_2) \\
  v_2 &= \frac{1}{\sqrt{2}}(V_1 - V_2)
\end{align*}$$

There is no reason it should be obvious to you why we make this transformation, but what the transformation is doing should be clear enough: we’ve chosen to manipulate the “DC” and “AC” components of the original variables. This makes (??) easier to manage, and your answer to the next few questions will demonstrate why this is so.

\(^1\)This is a very notation-intensive section, but don’t worry—take a deep breath, everything will be defined, and we will walk you through most of the algebra so you don’t have to spend too much time worrying about unimportant manipulations.

\(^2\) $N_1$ and $N_2$ are also mutually independent.
• Show that

\[
y_1 = \alpha (x_1 + n_1) + v_1
\]

\[
y_2 = \beta (x_2 + n_2) + v_2
\]

(9)
(10)

where \( \alpha = (A + B) \) and \( \beta = (A - B) \).

• Show that \( \langle n_1^2 \rangle = \langle n_2^2 \rangle = \eta^2 \) and that \( \langle n_1 n_2 \rangle = 0 \). Similarly, \( \langle v_1^2 \rangle = \langle v_2^2 \rangle = \nu^2 \) and \( \langle v_1 v_2 \rangle = 0 \).

• Show that \( \langle x_1 x_2 \rangle = 0 \). (See hint\(^3\).) Then show that \( \langle y_1 y_2 \rangle = 0 \).

We’ve completely de-coupled the two equations, simplifying future calculations. What coupled them previously, \( \Gamma^2 \), now appears through the following:

• Define \( \sigma_1^2 \equiv \langle x_1^2 \rangle \) and \( \sigma_2^2 \equiv \langle x_2^2 \rangle \). Show that \( \sigma_1^2 = \Sigma^2 + \Gamma^2 \) and that \( \sigma_2^2 = \Sigma^2 - \Gamma^2 \).

• We choose to place the constraint \( C \equiv \langle Y_1^2 \rangle + \langle Y_2^2 \rangle \) on the output to be minimized. Show that this constraint, which we will call \( C(\alpha, \beta) \) is now

\[
C(\alpha, \beta) \equiv \langle y_1^2 \rangle + \langle y_2^2 \rangle
\]

(11)

where

\[
\langle y_1^2 \rangle = \alpha^2 (\sigma_1^2 + \eta^2) + \nu^2
\]

\[
\langle y_2^2 \rangle = \beta^2 (\sigma_2^2 + \eta^2) + \nu^2
\]

(12)
(13)

**Breathing space**

So far all we’ve done is rewrite the equations. But note that since \( y_1 \) doesn’t depend on \( x_2 \), and \( y_2 \) doesn’t depend on \( x_1 \), we can now say that

\[
I(y; x) \equiv I(y_1, y_2; x_1, x_2) = I(y_1; x_1) + I(y_2; x_2)
\]

(14)

which is a nice simplification. Go ahead and show this if you like— or not if you don’t feel like it, and just take it on faith and common sense.

All right. Back to the work.

**Back to the work**

• Show that

\[
I(y_1; x_1) = \frac{1}{2} \log \frac{\alpha^2 (\sigma_1^2 + \eta^2) + \nu^2}{\alpha^2 \eta^2 + \nu^2}
\]

(15)

and from symmetry of the equations, find a similar solution for \( I(y_2; x_2) \) (note that this is obviously just a case of (??)).

So now we’re set. What we want to do is find the \( \alpha \) and \( \beta \) which maximize \( I(y; x) \) (equn. ????), and at mean time minimize \( C(\alpha, \beta) \) (equn. ???? — just like what we did in one-dimensional case. We define energy function:

\[
E \equiv C(\alpha, \beta) - \frac{k^2}{2} I(y; x)
\]

(16)

where \( k \) is constant, and the positive number \( \frac{k^2}{2} \) just sets the relative weight of maximizing \( I \) and minimizing \( C \).

\(^3\)Hint: remember, \( \langle X_1^2 \rangle = \langle X_2^2 \rangle = \Sigma^2 \).
The extremum we are looking for will be given by

\[
\frac{\partial E}{\partial \alpha} = 0 \quad (17)
\]

\[
\frac{\partial E}{\partial \beta} = 0 \quad (18)
\]

- Show that (17) reduces to

\[
2\alpha (\sigma_1^2 + \eta^2) - \frac{k^2}{2} \frac{\partial I(y_1; x_1)}{\partial \alpha} = 0
\]

Then show that this is the same quadratic equation in \( \alpha^2 \) (i.e., it involves \( \alpha^2 \) and \( \alpha^2 \)) as the one-dimensional case for \( A^2 \). If you plug through it, you will get to the solution to the quadratic equation,

\[
|\alpha| = \begin{cases} 
\sqrt{\frac{\nu^2}{\eta^2} \sqrt{1 + \frac{1 + k^2 \eta^2 / \sigma_1^2 \nu^2}{2(1 + \eta^2 / \sigma_1^2)}} - 1} \\
0, \text{if } \sqrt{\frac{\nu^2}{\eta^2} \sqrt{1 + \frac{1 + k^2 \eta^2 / \sigma_1^2 \nu^2}{2(1 + \eta^2 / \sigma_1^2)}} - 1} \text{ is not real}
\end{cases}
\]  

(20)

Similarly for \( \beta \),

\[
|\beta| = \begin{cases} 
\sqrt{\frac{\nu^2}{\eta^2} \sqrt{1 + \frac{1 + k^2 \eta^2 / \sigma_2^2 \nu^2}{2(1 + \eta^2 / \sigma_2^2)}} - 1} \\
0, \text{if } \sqrt{\frac{\nu^2}{\eta^2} \sqrt{1 + \frac{1 + k^2 \eta^2 / \sigma_2^2 \nu^2}{2(1 + \eta^2 / \sigma_2^2)}} - 1} \text{ is not real}
\end{cases}
\]  

(21)

Some final comments: the variable transformation to lower-case variables that you did was actually a Fourier transform. For matrices of the form \( S_{ij} = S(i - j) \) (called Toeplitz, or cyclic matrices), Fourier modes of \( S(i - j) \) are actually eigenvectors of \( S \), with the Fourier components being the eigenvalues. Note that the covariance matrix of \( X \) is a Toeplitz matrix. Fourier transforming, just like we did here, decouples the equations in any number of dimensions. So the solution you found is actually the solution to the \( N \)-dimensional problem, not just the two-dimensional problem.