Computational Perception
15-485/785

Assignment 3: Coding

Due: Tuesday, Feb. 26

1 Optimizing the gain

Shannon's source coding theorem says the lower bound on the average code length (in bits) to encode a random variable is given by its entropy. Although the entropy provides a lower bound for the length of the optimal code, it does not say how this could be constructed. Even after 50 years of research, the question of how to design source codes that achieve the lower bound is the subject of open research and entire graduate courses. So, here we will just try to understand why the problem is hard. As usual, the necessary files are availalbe on the course website.

1. (*5 points*) First, generate a random distribution containing 5000 samples from a uniform distribution using the MATLAB function, `y = rand(5000,1).`

   `[n,x] = hist(y,256)` bins the samples `y` into 256 evenly spaced buckets (the coordinate of the bin centers are given by `x`) and returns counts of the number of elements falling in each bucket by `n`. (`hist(y,256)` alone plots the histogram.) Given a vector of such counts, write a simple function to estimate the entropy of the distribution:

   $$\sum_v p(v) \log_2 p(v),$$

   where v is the random variable of interest. The function should take as input an image and the number of bins to use and also handle bins where $p(v) = 0$. What entropy do you estimate for `y`?

2. (*3 points*) Changing the number of bins will change the entropy estimate. Explain what is going on. What is happening when we make the number of bins arbitrarily large?

3. (*5 points*) Drawing another random sample will also change the entropy. Explain what is going on. How could the true lower bound on estimated code length be obtained?

4. (*3 points*) The file we provided `images.mat` contains two images. You can load them with `load images.mat` after which the variable `Xnat` will contain an image of a natural scene, and the variable `Xtxt` will contain an screenshot of last year's version of this assignment. You can view them with the MATLAB function `imagesc`. The command `colormap gray, axis image off` will make it look better.

Consider the distribution of intensity values taken on by pixels in `Xnat`. Plot a histogram of the pixel values of this image using 256 bins, and compute its entropy using the MATLAB function you just made above.

5. (*3+3 points*) Similarly, plot the histogram and compute the entropy of `Xtxt`. Why is the entropy of `Xtxt` so much lower than that of `Xnat`?

6. (*5 points*) Suppose you want to compress an image and transmit it through a channel of limited capacity, say 2 bits per pixel. One way to do it is to treat the image as a one-dimensional data stream. Assume the original precision of the data is 8 bits. Which set of images could we transmit perfectly through the channel? Explain why just in one sentence (we will explore this in detail in the following questions).

7. (*3 points*) The MATLAB function `ecdf` computes the empirical cumulative distribution function. The result is essentially the integral of the histogram of the image. Use the function to plot the cumulative distribution of `Xnat` and `Xtxt` (You will need to reshape the image variable to a column vector.) Plot the cumulative distribution for each image.

8. (*3 points*) We want to transmit each image separately. In each case we only have a capacity of 2 bits per pixel. Describe how you could use the ecdf to design a coder. For the `Xnat` image, are you making full use of the channel's capacity? Explain why.

9. (*3 points*) For the `Xtxt` image, does this coding scheme transmit the images without error? Explain why.

10. (*bonus points*) Implement your coder and measure its performance in terms of the reconstruction error.

2 Optimizing the basis functions

Now we're going to move to a similar but much higher dimensional problem: choosing basis functions to represent images. Here we are interested in the redundancy (e.g., correlation) among different pixels, not in the redundancy of pixel values that we just explored in the question 1. We'll use a simple, linear formalism, representing patches of images as the weighted sum of a number of basis functions, which are themselves image patches of the same size. For the usual, pixel-domain representation of an image, the basis functions are just blocks of black pixels with a single white pixel lit somewhere. Since real images are redundant (adjacent pixels are correlated), a better representation should have more spatial extent, incorporating the sources' correlational structure in the basis functions. In order to allow ourselves to use the notation of linear algebra, we'll unroll the pixel values of each 8×8 image patch into a vector of length 64. Then we can write

$$\mathbf{x} = \mathbf{A}\mathbf{s} \tag{1}$$

where the column vector $\mathbf{x}$ is the original image patch, each column of the matrix $\mathbf{A}$ is a basis vector, and the column vector $\mathbf{s}$ contains the coefficients corresponding to each basis vector in the image patch's representation in the basis. This equation describes how to reconstruct an image from its representation $\mathbf{s}$ in the basis described by $\mathbf{A}$. Assuming the matrix $\mathbf{A}$ is invertible – that is, the basis is complete – an image can be encoded into the basis using

$$\mathbf{A}^{-1}\mathbf{x} = \mathbf{s} \tag{2}$$

and so, insofar as the operation is reversible, any such code works fine. However, there are a number of ways in which some bases can still be better than others, which we will explore here with some simple exercises in image compression.

1. (*8 points*) One important basis is the discrete cosine transform, which is used in JPEG compression. MATLAB provides the functions `dct2` and `idct2` to encode and decode data in a DCT basis. (These are in the image processing toolbox, so you may not have them if you're working at home.) Apart from the fact that these functions expect the image data in a matrix form rather than unrolled as a vector, they can be described using the linear formalism above: `dct2` implements equation (1) and idct2 implements equation (2). Given what you know about linear systems, how can you extract the DCT basis using the MATLAB DCT functions? Recover all 64 basis functions for DCT on 8×8 matrices. Store it into a 64×64 matrix called `Adct` where each column contains an unrolled DCT basis. Hint: You may find the commands `eye` and `reshape` useful.

   What you should turn in:

   - Your code to generate `Adct`.
   - Visualize `Adct` using the provided MATLAB code `plotBFs(A)`.

2. (*3 points*) Now write (very simple) MATLAB functions that use equations (2) and (1) to encode and decode an image $\mathbf{X}$ using some basis $\mathbf{A}$. We provide the functions `img2vec` and `vec2img` which take care of converting the image back and forth from a square matrix into a series of column vectors representing unwrapped 8×8 pixel patches; you need just convert each column to and from the basis. If `xx = img2vec(X)` then you can extract the ith vector as `xx(:,i)`. Use `pinv` to perform the matrix inversion in equation (1), because if the matrix is noninvertible it chooses a "pseudo-inverse" that minimizes the sum squared error on reconstruction.

3. (*3 points*) The file `bases.mat` contains a number of 8×8 bases, which you can view with the command `plotBFs(A)`:

   - `Anat` is a basis optimized using ICA to represent natural image data like `Xnat`.
   - `Atxt` is a basis optimized using ICA to represent textual images like `Xtxt`.

   Using your encoding function, encode `Xnat` with the basis function `Anat`. Now let's examine the distribution of the coefficients $p(s)$. Plot the histogram of the coefficients and compute the entropy as you did above. (We can examine each coefficient (associated with different basis function) separately, or just mix them together to examine the characteristics of the average. In this assignment, please submit the results by mixing all the coefficients.)

4. (*5 points*) Say you wanted to quantize the real-valued pixel intensities or `Anat` basis coefficients in order to store them in the digital device. Does the form of $p(s)$ suggest a strategy for compressing the image? In particular, how (generally) should you choose bit strings to represent the quantized value of basis coefficients s, given what you know about $p(s)$? Can you do the same thing as effectively with the raw pixels x?

5. (*3 points*) `Xnat` and `Xtxt` were optimized using independent component analysis (ICA) to represent two different sets of data. As we briefly discussed in the class, ICA selects a basis by gradient ascent on the likelihood of the data given the model, $p(x|\mathbf{A}, s)$ under the assumption that the basis coefficients s are independent and distributed sparsely (see the reading Lewicki (2002) for more details).

   View the two bases with `plotBFs`. How do they differ? What in the structure in the images of text in contrast to natural images do you suppose accounts for the differences?

6. (*bonus points*) Compare the efficiency and accuracy of different code.

3 Redundancy in the natural image

Daniel Kersten conducted an experiment in which subjects were presented with images that had a number of random noise (see Fig. 8.2 in the handout Wandell (1995), Ch. 8). The task of the subjects was to guess the correct values of the pixels as accurately as they could. Kersten found that subjects could do this task with very high accuracy.

1. (*5 points*) What does this result indicate about the statistical properties about images?

2. (*5 points*) What does this say about the use of pixels for an image basis?

Brian Wandell suggested a variation on this experiment which was instead having subjects correct the pixel values, to have subjects reconstruct the basis function coefficients of different bases. In the case of a pixel basis, this is simply the pixel value itself.

3. (*5 points*) For what types of basis functions would you expect subjects to perform well, and poorly? In your answer be sure to describe what effect the random coefficients would have on the image.

4. (*10 points*) How would you expect subjects to perform on an ideal (in the sense of coding efficiency) basis? What advantages would this type of code have for image compression?

4 Designing perceptually relevant cost functions

A simple metric to describe how close a reconstruction from compressed data is to the original is to look at the mean squared error (MSE) of the residuals (the original minus the reconstructed data). This can be expressed in terms of dB signal to noise ratio or the percentage error of the reconstruction. For applications like image or sound compression, however, the MSE does not correspond well to the perceived quality.

1. (*5 points*) Given what you know about human auditory sensitivity, why would you expect this?

2. (*5 points*) Describe a method that could be used to obtain a more accurate (but still objective) metric of perceived sound quality. Your answer should describe the goals of the method in addition to the method itself.

3. (*5 points*) Give a quantitative statement of the method in the previous answer in terms of an equation. Feel free to define functions which you might not know, but could be measured or looked up. Be sure to explain what each term means.

Note that you can extend the same idea to the image coding given the human visual sensitivity, or even to the coding of the other sensory modalities.

5 Seeing through different spectra

The response of a photoreceptor is determined by its spectral sensitivity function. This is usually expressed as the photon absorption efficiency as a function of wavelength. For a photoreceptor of class i at the retinal location (x, y) its response is given by the inner product of ϕ_i with the light spectrum at that location,

$$c_i = \int \phi_i(\lambda) I(\lambda) d\lambda \,. \tag{3}$$

For simplicity, let us assume that at each location we have three photoreceptors.

1. (*15 points*) Suppose we are looking at a display with three photoelements (e.g., pixels on an LED display). Write down the conditions that the pixel values need to satisfy in order to appear perceptually identical to the real stimulus at a given location. You need to define your variables under your own assumptions (e.g., how to get three pixel values and how to represent it; here let's assume that the systems are all linear.)

It is known that different people have different spectral sensitivity curves, due to the fact that there are different alleles (varieties) of the L and M cones. For example, in people with normal color vision, the M and L cones have a peak sensitivity around 530 nm and 560 nm, respectivity, giving a spectral difference of 30 nm. Alleles of the M and L pigment, however, can result in spectral differences of just 3-12 nm, resulting a greatly reduced red-green sensitivity. This occurs in about 8-10% of the US male population.

2. (*15 points*) Let $\Phi = \{\phi_1, \phi_2, \phi_3\}$ and $\mathbf{c} = \{c_1, c_2, c_3\}$. Given two sets of spectral sensitivity curves, Φ_a and Φ_b, derive a relation that would allow us to transform the pixel color such that the response $\mathbf{c}_a$ is the same as the response in $\mathbf{c}_b$. Again, state your assumptions and define your variables.