Computational Perception
15-485/785

Assignment 5

due Tuesday, April 22

This assignment is different in character from the previous assignments you have done in that it focused entirely on your understanding and interpretation of the reading assignments handed out in class in addition to the material presented in the lectures. There is no programming component. These questions are necessarily open-ended and do not always admit a unique answer, so I have tried to provide a brief statement of what is expected in the answer. As a general rule, your answers should be concise explanations in the sense that it could sufficiently answer the question and explain the relevant issues to someone who is not familiar with them. Refer to the example answer given in the first assignment for guidance.

1 Readings

1. Adelson, E. H. (2000). Lightness perception and lightness illusions. In Gazzaniga, M., editor, *The New Cognitive Neurosciences*. pages 339-351. MIT Press, Cambridge.

2. Mingolla, E. and Todd, J. T. (1986). Perception of solid shape from shading. *Biol. Cybern.*, 53:137-151.

2 The lightness equation

The general problem of how the brain is able to accurately perceive the real-world intrinsic properties of objects and surfaces is called perceptual constancy. An important example of this is the ability of the visual system to accurately perceive external surface properties and 3-dimensional structure from the 2-dimensional intensity patterns on the retinal image, or, in the case of binocular perception, retinal images. Deriving shape from shading plays a primary role in our perception of surfaces. The complementary problem of shape from shading is the perception of lightness. Here, rather than use visual cues such as shading

patterns to infer surface shape, the problem is how to accurately perceive the surface properties such as the lightness of a surface. Both involves the integration of a many different types of visual cues and cognitive knowledge. More generally, surface perception also includes properties such as color, texture, and material, all of which the human visual system can accurately perceive under a wide range of conditions.

Most approaches to this problem start from a physical model of light reflection, where surface lightness $L(x, y)$ is a product of the illuminance image $E(x, y)$ and a reflectance image $R(x, y)$. This can be generalized to allow for dependencies of the (two-dimensional) angle of the incident light, ϕ_i, and the angle, ϕ_e, of the emitted light

$$L(x, y, \phi_e) = E(x, y, \phi_i)R(x, y, \phi_i, \phi_e).$$

Although this equation is an approximation, it is still possible to draw from it a number of insights.

1. Why is the color 'white' a perceptual category and not a physical property?

   Your answer should make use of the lightness equation and also give an example of a perceptual stimulus and conditions that would demonstrate this phenomenon convincingly. Remember that your answers should be *explanations* and not simply a citation or unexplained reference.

2. How is it possible to solve this equation?

   List at least two ways in which prior knowledge could be used to obtain a solution to this equation which is an example of an *ill-posed* problem.

3. How is this equation an idealization of realistic object properties and lighting conditions?

   Your answer should contain examples of objects, surfaces, and conditions that would be different to capture with this equation. You should also explain why this makes the problem more difficult.

4. What generalization of this equation did Adelson in the 'Lightness Perception and Illusions' use to account for a broader range of perceptual phenomena? Does this make the problem more or less tractable?

   You answer should explain the generalization, the idea behind it, i.e. how it overcomes some limitations of the standard equation, and what types of visual stimuli or environments it applies to.

3 Local and global cues

The lightness equation describes the approximate relationship between the surface orientation and the amount of radiant light reflected to the observer. With the appropriate assumptions, it is possible to work out the local orientation of the surface. Local cues (both point intensities and local visual features), however, do not always accurately convey the true underlying surface properties. Shape from shading algorithms, such as those surveyed in the Zhang, Tsai, Cryer, and Shah paper, usually impose some type of regularizer, such as a smoothness constraint, to resolve the ambiguity between local cues and the global percept.

1. Cite and explain two examples that show why local intensity cannot explain the perceptual interpretation of 1) lightness and 2) surface structure.

   Your answer should explain what local cues are available and why they fail to provide unambiguous information about the global percept.

2. Using the arguments from the Edelman paper 'Representation is Representation of Similarities', what criticisms could be made about the types of models used in the shape from shading survey paper?

   Your answer should summarize the basic idea of shape representation put forth in the Edelman paper and explain how it differs from the the models of the survey paper.

4 Psychological validity

Using the inverse graphics approach to the computation of surface structure often depends on making several assumptions about what the visual system knows or can infer from the visual scene. The paper by Mingolla and Todd on the 'Perception of Solid Shape from Shading' described experiments that assessed subjects ability to judge slants and tilts on various computer generated ellipses.

1. What evidence did Mingolla and Todd give to argue that the assumptions of Lambertian reflectance and known illuminant direction are not accurate models for how the visual system estimates shape?

2. The authors also argued against the assumption that global shape is computed from local estimates of surface orientation. Summarize their argument and evidence.

3. In that paper, computer generated ellipses were used to estimate subjects' perception. Why were these chosen and what issues would you consider for more general types of shapes?

5 Alternative formulations

The most straightforward way to formulate the problem of surface structure inference is in terms of inverse graphics. This formulation provides well defined (if ill-posed) equations for the solution. Some of the articles have questioned, given what is known about visual perception, whether this is the right problem to solve. In other words, the problem might be well-defined, but it might not be necessary to solve it in order to perform higher level vision problems like object recognition or scene analysis. This leads to the question of what alternative formulations of the basis perceptual problem might exist.

1. Give one example of a computational goal that simplifies the problem perception of the perception of surfaces.

   Your answer does not need to be specific in terms of mathematical formulation, but it should describe a specific idea and explain how it overcomes one or more limitations or unnecessary properties of the inverse graphics approach. You should also explain the limitations and/or why they are not necessary for solutions to other problems in computational vision that inference of surface structure might subserve. For your answer, you may use ideas from the readings or of your own.

6 Sources, code, and blind source separation (50 pts)

In this problem you will implement a maximum likelihood blind source separation (BSS) algorithm. Although the focus will on BSS, the problem draws on ideas discussed throughout the course. This question involves some (light) programming and the use of matlab. You should turn in all of your code (which isn't much – you should be able to do everything in two functions). A soundcard and headphones aren't absolutely necessary, but it will make it more fun. You can use the matlab function `wavwrite` to write `.wav` files so that they can be played in an external player. The data files and helper functions needed for this question are available on the course weg page.

The basic model for the problem of blind source separation is that the sources, $\mathbf{s}(t)$ are mixed instantaneously onto the mixtures $\mathbf{x}(t)$ through the mixing matrix $\mathbf{A}$

$$\mathbf{x}(t) = \mathbf{A}\mathbf{s}(t)$$

Each vector $\mathbf{x}(t)$ represents the values of each mixture at time t. The likelihood for a given vector $\mathbf{x}$, is

$$\frac{P(\mathbf{s})}{|\det \mathbf{A}|}$$

where $\mathbf{s} = \mathbf{A}^{-1}\mathbf{x}$. It will be convenient to operate on blocks of mixture vectors, so we can also write $\mathbf{S} = \mathbf{A}^{-1}\mathbf{X}$, where $\mathbf{S}$ is a $2 \times N$ matrix with the sources given in each row.

The independence assumption means that $P(\mathbf{s}) = \prod_m P(s_m)$. Here we will make the simple assumption that all the sources have a Laplacian distribution, $P(s_m) \propto \exp(-|s_m|)$.

We assume that the samples are independent which means the likelihood of a pattern ensemble is given by

$$P(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N|\mathbf{A}) = \prod_k P(\mathbf{x}_k|\mathbf{A})$$

Since we are implementing a maximum likelihood algorithm, we need a function to evaluate the likelihood given a set of data. This function is not strictly necessarily, but it is an invaluable debugging tool that will help you make sure your algorithm is correct.

1. Write a function, `loglike(A,S)`, that calculates the log likelihood for a mixing matrix $\mathbf{A}$ and source matrix $\mathbf{S}$. Since we are just using this to ensure that we are doing proper gradient ascent, you can ignore any constant factors. A useful fact is that the determinant, $\det \mathbf{A}$, is equal to the product of the eigenvalues of $\mathbf{A}$, which can be calculated using the matlab function `eig`. Note that you have to pay attention to numerical stability, which is why we compute the *log* likelihood.

To maximize the (log) likelihood, we compute its gradient

$$\begin{aligned} \Delta\mathbf{A} &\propto \mathbf{A}\mathbf{A}^T \frac{\partial}{\partial \mathbf{A}} \log P(\mathbf{x}|\mathbf{A}) \\ &= -\mathbf{A}(\mathbf{z}\mathbf{s}^T + \mathbf{I}) \,, \end{aligned}$$

where $\mathbf{z} = (\log P(\mathbf{s}))'$. In the case of the Laplacian, $\mathbf{z} = -\operatorname{sign}(\mathbf{s})$. Here we use Amari's rule which scales the gradient by $\mathbf{A}\mathbf{A}^T$. This yields the so-called natural gradient yields a simple equation and greatly speeds convergence.

One approach would be to calculate the gradient for each mixture vector $\mathbf{x}_k$, update the gradient, and iterate. It is computationally much more efficient, however, to take advantage of Matlab's highly optimized matrix routines. Instead what we can do is compute the sum (or average) gradient for a whole block of mixture vectors, $\mathbf{X}$, with a just single line of code.

2. Convert the gradient step to block form, i.e. show that

$$\begin{aligned} \Delta\mathbf{A} &\propto \mathbf{A}\mathbf{A}^T \frac{\partial}{\partial \mathbf{A}} \log P(\mathbf{X}|\mathbf{A}) \\ &= -\mathbf{A}\mathbf{Z}\mathbf{S}^T - n\mathbf{A} \,, \end{aligned}$$

   where n is the number of vectors in the block. Note that if you find yourself spending too much time on this problem, you can simply use the result on go on.

After each block, the mixing matrix $\mathbf{A}$ is updated according to

$$\mathbf{A} = \mathbf{A} + \epsilon\Delta\mathbf{A}$$

Now we will get on to the implementation of the algorithm. In this problem, you have been given two mixtures in the file `X.mat`. The problem is to infer the original sources.

You should generate the mixed waveforms `x1.wav` and `x2.wav` using `wavwrite`, so you can hear what the mixtures are like and to make sure you understand the data format. The sampling frequency for both files is 22050 and the number of bits per sample is 16. You will also need to use the provided function `soundnorm` to scale the waveforms to fit in the $[-1, 1]$ range of the `.wav` format.

3. Implement the an algorithm to perform blind source separation. Your code should consist of just two files, `loglike.m` from above, and the algorithm itself in `bss.m`. Be sure to make your code clear and use comments so I can figure out what you're trying to do and give you partial credit even if your code doesn't give the right solution.

   The algorithm involves the following steps:

Step 1: Load the mixtures and choose initial values for $\mathbf{A}$ and ϵ. You can use the identity matrix for $\mathbf{A}$. A useful way to set the initial stepsize is to start small, e.g. 0.001, and increase it until you see a steady increase in the log likelihood over the first 1000 iterations.

Step 2: Select a random set of samples. Start with blocksize of 1000. (Hint: use of the provided function `unidrnd` and matlab indexing will be helpful here, e.g. if `r` is a random index array, `x[r]` is a vector with only those elements.)

Step 3: Estimate the sources given the current mixing matrix.

Step 4: Calculate the gradient step $\Delta\mathbf{A}$ and update the value of $\mathbf{A}$

Step 5: Calculate the new data log likelihood.

Step 6: Every 100 iterations (or so), update a plot of the log likelihood and plot the new basis vectors. This will greatly speed up debugging any errors in your code. To make a running graph, you can use code like the following

```
figure(1);
plot(i,logL,'+');
hold on;
drawnow;
```

where `i` is the iteration number. The function `plotA` is provided to plot the basis vectors.

Step 7: Loop to step 2 and repeat until solution is reached. the number of iterations needed depends on your stepsize, but you will probably need between 1,000 and 5,000. To converge to a solution, you will also need to reduce the stepsize. This effectively averages the gradient step over a large set of patterns. A simple strategy here is to reduce the stepsize by a factor of 2 every 1000 iterations.

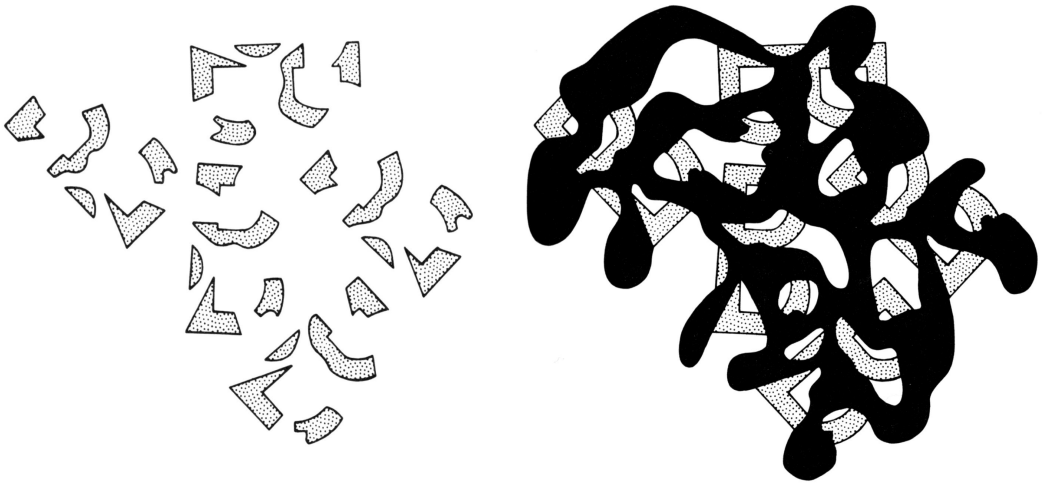
4. What values do you get for the mixing matrix $\mathbf{A}$? Are they consistent for different runs? You solution should also include the plots of the log likelihood curves and the basis vectors.

5. Use the provided function `snr` to calculate the signal to noise ratio (in dB) between the original sources and the reconstructed sources. The original sources are provided in the files `bach.wav` and `speech.wav`. It is also helpful to normalize the sounds to the range $[-1, 1]$ using the provided function `soundnorm`. This minimizes scaling differences and prevents clipping of the `.wav` files. How do these values compare to your subjective impressions of the reconstructed sound?

6. In step 2 of the algorithm, you drew a random set of samples from the waveform. Explain why was this possible and perhaps even advantageous. What does this reveal about both the limitations and the generality of the BSS algorithm?

7 Perceptual Organization: one from many (30 pts)

Recall the example of Bregman's B's:



1. Describe how this phenomenon is similar to perceptual completion of auditory signals as in the picket fence effect. In particular, state why the addition of the ink (or noise) aids the perceptual interpretation of the patterns. What attentional processes are involved in the perception of these patterns?

2. Give a quantitative argument for how the addition of the foreground information simplifies the problem. Hint: One way to answer this is to frame the problem in terms of a search space and show that the additional information reduces the size of that space.

3. Theories of auditory and visual grouping hypothesize many types of cues, such as common covariation, separation, etc. Select two such cues, one from each modality, and describe some of the difficulties that are likely to arise in applying the abstract idealizations to natural signals.

4. Now consider two models we discussed in lecture: the auditory stream analysis model of Cooke and the selective attention model of Itti and Koch. Both of these are examples of bottom models of perceptual organization, but we know of several examples where top-down knowledge has a profound effect on our interpretation of the scene. For each of these models, propose some ideas for how they could incorporate top-down knowledge. Your answer should explain the benefits and limitations of each idea in terms of each model. You should also discuss what difficulties there might be in the integration, e.g. what types of knowledge would be easy to integrate with the existing model and what types would be difficult.