Optimization
Project 1 – Linear Programming

Deliverables
One well-written pdf file and one Python code file (.py or .ipynb), submitted to Canvas. Your report should go into some detail about how you solved the problem, include some graphs that explain your results, and include relevant code chunks in the final output. Your report can be created by taking screenshots of the code/graphs and assembling it in a word document, then export as a pdf file.

Problem Description
Image processing is an important step in many large-scale data science projects. In this project, you will build a tool that separates the background of an image from the foreground. To do this we will pose the segmentation problem as a linear program and solve it.

Imagine an image as a 2D grid of pixel intensities (let's work only with greyscale images). We would like to transform this grid of intensities into a network. A network consists of nodes and connections/links. The nodes themselves do not have any numeric value, but the connections between the nodes do. We would like to represent each pixel as a node, and the similarity between neighboring pixel values to be the numeric value associated with the connection between neighbors. Hopefully then, the similarity between neighboring pixels that represent the border between the background and foreground is quite small. In a picture of me against a greenscreen, the similarity between the last pixel of my face and the first pixel of the greenscreen is pretty small. If we know the location of one pixel that is in the background, and one pixel that is in the foreground, we want to find which connections in the network could we sever that add up to the lowest sum of similarities and would completely separate the one background pixel from the one foreground pixel. This is what we will do for the project!

This seems like an integer program, but it turns out there is a nice result from mathematics called the Max Flow / Min Cut Theorem that allows us to reformulate this problem as a completely different linear program. If we think of each connection in the network as a water pipe where the maximum flow rate through the pipe is equal to the similarity from above, and each node is a joint between pipes, we can, in fact, solve the cutting problem by instead maximizing the total flow of water through the network from the one background pixel to the one foreground pixel. It is absolutely WILD that this works! On top of this, the max flow problem is easier to solve than the original cutting problem!

In this project you will use a few photos to explore the max flow / min cut theorem. One photo will be provided for you, and the others will be created by you.

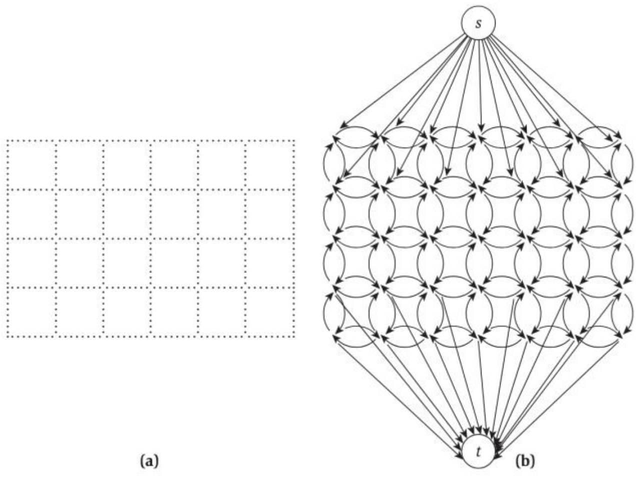
Specifics

1) *Create the network*: A picture is merely a grid of pixels values. Let's say we are working with a 20x20 pixel image. There are then 400 numbers that represent the intensity/brightness of each pixel. To represent this image as a network we create a 400x400 matrix. The $(i,j)^{th}$ entry of the matrix represents the similarity between pixel i and pixel j. Here pixel i, for example, is at a particular row and column of the image. For simplicity we will assume that the $(i,j)^{th}$ entry of the network matrix is the same as the $(j,i)^{th}$ entry – one represents flow from i to j, and one represents flow from j to i. In this example we will only calculate the similarity between neighboring pixels, and assume non-neighboring pixels are not connected (connection value

equal to zero). It is up to you how you decide what pixels are neighbors – you can use just the pixels above/below and left/right of each other, or you can add in the diagonally adjacent pixels too. A pixel should not be considered a neighbor of itself.

The similarity of two pixels should be calculated as $100\exp\left(-\frac{(I_i-I_j)^2}{2\sigma^2}\right)$, where I_i is the intensity of pixel i, and I_j is the intensity of pixel j. This similarity should be rounded UP to the nearest integer. You will need to experiment to find the value of σ that works well for you. In the image attached to this project $\sigma = 0.05$ worked well for me, but this may or may not work well for other images.

In addition to the pixels representing nodes on the network, we need two more nodes – called terminal nodes. One terminal node is called the source node, and one terminal node is called the sink node. The source node is ONLY connected to the one pixel in the background, and water can ONLY flow from the source node to the background pixel (not in reverse), and the sink node is only connected to the one pixel in the foreground, and water can ONLY flow from the foreground pixel to the sink node. The max flow rate for both connections should be equal to the largest similarity of all the connections between pixels. This means that for a 20x20 pixel image, the network is represented by a 402x402 matrix. It is up to you to manually decide which pixel in the image is in the foreground, and which is in the background.

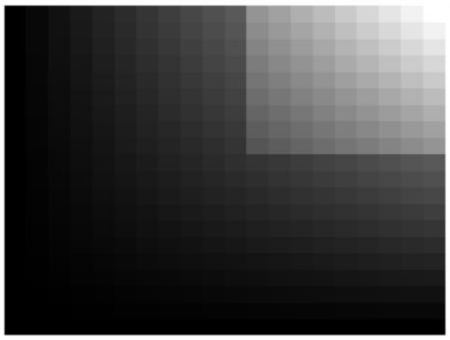
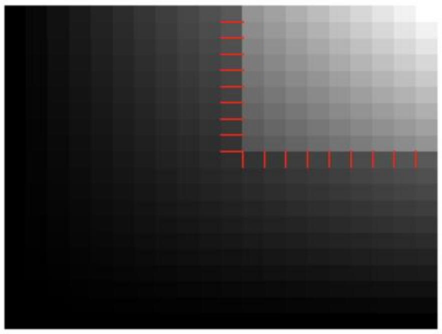
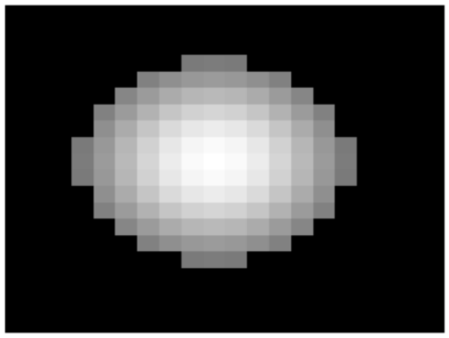
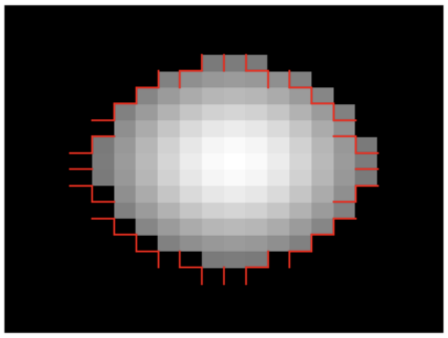


The above image illustrates the conversion of an image into a network. Each arrow represents the connection between pixels. I took this image from a text book example that connects flow from the source node (s) to every pixel in the image, and every pixel to the sink node (t). In our example water only flows from the source node to one pixel and from one other pixel to the sink node.

HINT: if a pixels is on row R and column C (indexed based on 0) of an image, then if you rearrange the 20x20 pixels into a vertical array of 400 numbers, that pixel can be found to be at the R(number of columns)+C location of the array.*

2) *Formulate the LP*: Our goal is to maximize the flow from the source node to the sink node. To do this we will form decision variables that correspond to every non-zero link in the matrix. Each decision variable is bounded above by the max flow rate along that link, and bounded below by zero. The network tells us how much flow is possible between each node, but the decision variables tell us how much water actually flows between each node. A 10 gallon per minute pipe connected to a 1 gallon per minute pipe can only flow 1 gallon per minute. The objective is to maximize the flow out of the source – just the decision variable from the source to the background pixel. And there are constraints for every pixel node on the network that represent that the sum of flow into each node (sum of all decision variables going into a pixel) is equal to the flow out of each node (sum of all decision variables going out of a pixel). After formulating the LP, solve it! What is the optimal objective value?
3) *Find the cuts*: Going from the solution of the max flow problem to the actual cuts we are interested in is a little tricky. First calculate the residual network. This is a network between the same nodes as the original nodes, but the value of each connection is equal to the maximum possible flow between the nodes minus the actual flow between the nodes (the optimal decision variables from the LP above). Perform a depth-first search from the source node of this residual network (https://en.wikipedia.org/wiki/Depth-first_search) to find all the nodes that are accessible from the source node – some connections in the residual network will be zero that were not zero in the original network. If a pipe flows at capacity then these connections are severed in the residual network. Put all accessible nodes in the residual network in one group, and all inaccessible nodes in another group. Any link on the original network between a node in the accessible group and the inaccessible group should be cut! Add up the max flow rates (similarity between pixels) of all cut pipes. What is this equal to? How is this related to the objective value from the LP?

   I did this for the image attached to this project and found the following optimal cuts.

4) You may find it easier to formulate this problem if you create decision variable for EVERY possible link in the network but put an upper bound of zero for each non-connected pair of nodes on the network. If we have a 20x20 image, then we have a 402x402 matrix representing the network, which has 161,604 values in it, and thus you can formulate an LP with 161,604 decision variables. MOST of these decision variables should be forced to zero, using the upper bound argument, though. If you do this with gurobi, you can run:

```
mod=gp.Model()

flow = mod.addMVar((402,402),ub=network)
```

Formulating the problem this way is easier to code but will take longer to solve than if you just create variables for each non-zero link in the network. It's up to you how you solve the problem. Solving the LP only requires 3 more lines of code. One to set the objective, one to set all the flow conservation constraints (flow into and out of each pixel must be the same- addConstrs!), and one to optimize it.

5) Your python code should be clear and easy to read. The first line should load a csv file that will represent the pixel intensities for the image that is to be segmented. The next few lines should specify which row/column the single background and foreground pixels are in, and what sigma is equal to. When your assignment is graded it will be evaluated using a new image. The output of your code should be the image with the cuts drawn in red on it. A useful tool for this is matplotlib's pcolor function.
6) Write a pdf file that analyzes this max flow / min cut theorem. Find the optimal cuts for the image attached here and find a few simple / small images that you can cut too. Pretend that you work for a startup that is competing with PhotoShop. Your boss wants an image segmentation tool in the software and has asked you to explore the possibility of using LP to do this. Your boss is familiar with optimization and the max flow /min cut theorem but has not tried it on image segmentation before. Include pictures, graphs, code chunks, and whatever else you believe will be useful to make a recommendation to your boss.
7) Grading will be based 66% on whether you get the right answer or not when we re-run your analysis with new data. If you don't get the right answer or your python code file doesn't run, we will go through your code and give partial credit accordingly. The easier it is to read your code the easier it is for us to understand what you're doing, so use a lot of comments in your code! The remaining 34% of your grade is based on the quality of your analysis and presentation of results in the pdf file. Write this as if you were actually going to submit it to your boss.