Graduate Artificial Intelligence 15-780

Homework 5: *Perception, Computational Social Choice and Computational Game Theory*

Out on April 13rd
Due on May 2nd

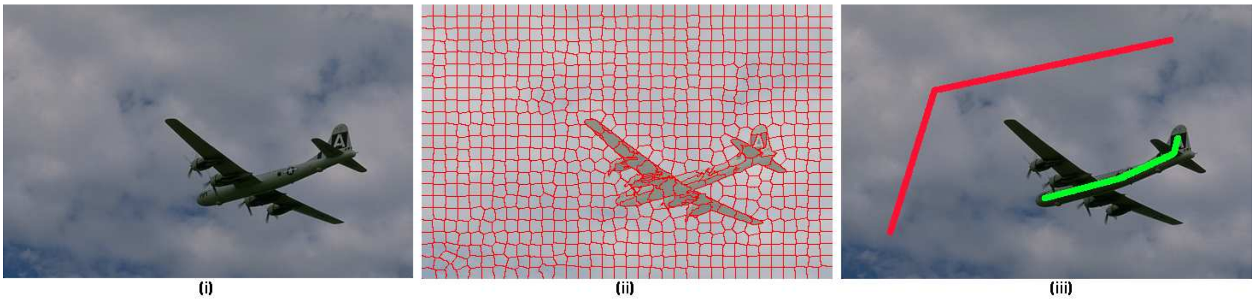

Figure 1: Example of image (i), its superpixels segmentation (ii) and the user-provided scribbles (iii). The red scribble represents background and the green one foreground.

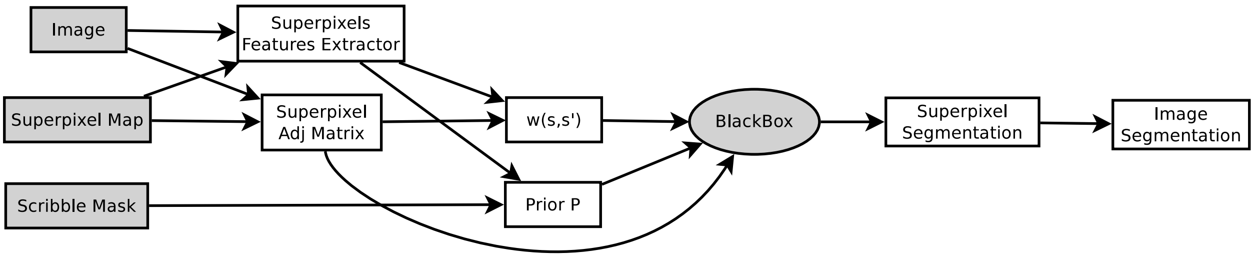

Figure 2: Summary of the pipeline that you will implement in this problem. Grey items are provided.

Problem 1: Assisted Background/Foreground Segmentation [Felipe, 70 pts][1]

Given an image of $h \times w$ pixels, we want to decide, for each pixel x_i, if it is background (i.e., $y_i = 0$) or foreground (i.e., $y_i = 1$). In this problem, you will implement a program that performs this task with a little help from the user. This help is in the form of scribbles or strokes on the image and their respective labels. Formally, for each image, two sets are also provided, L_{bg} and L_{fg}, such that all pixels $i \in L_{\text{bg}}$ are labeled as background by the user (similarly for L_{fg}).

One limitation of this approach is the number of random variables represented by the image. For instance, an image of 1024 by 1024 pixels has 2^{20}, i.e., $x_0, \ldots, x_{2^{20}-1}$, and we want to decide if x_i equals 0 or 1 for all x_i. A common approach to overcome this problem is to consider superpixels, that is, small segments of the image in which all the pixels are "similar". Figure 1 shows an example of image and its superpixels.

For all the images considered in this problem, a mapping from each pixel i to the corresponding superpixel $\mathbf{s}$ is provided. We also provide a blackbox to perform the inference over the superpixels.

Your pipeline

Figure 2 presents the overall view of your pipeline.

(a) **Image**: images are provided as JPEG files.

(b) **Superpixel Map**: one ASCII file for each image. The first line in a superpixel map file is "`h w S`" where h (w) is the height (width) of the image and S is the total number of superpixels. The file contains h more lines and each line has w columns (separated by one blank space each). The value at line i and column j represents the superpixel id of the pixel at position (i, j). The superpixel id is in $\{0, \ldots, S-1\}$.

(c) **Scribble Mask**: Same format as the Superpixel Map, except that S is omitted in the first line and the value at position (i, j) is either: 0 for background; 1 for foreground; and 2 for unknown.

[1] Inspired in a problem by Dhruv Batra.

(d) **Superpixel Feature Extractor**: given a superpixel $\mathbf{s}$, this function maps all set of pixels in $\mathbf{s}$ to the features $x_{\mathbf{s}}$ used by your pipeline.

(e) **Superpixel Adjacency Matrix**: adjacency matrix of the graph defined over the superpixel space. Formally, this is the undirected graph $G = (V, E)$ where $V = \{0, \ldots, S-1\}$ and $E = \{(\mathbf{s}, \mathbf{s}')|$ pixels $i \in \mathbf{s}$ and $j \in \mathbf{s}'$ are neighbors in the image$\}$.

(f) $P_{\mathbf{fg}}(x_{\mathbf{s}})$: the probability the superpixel $\mathbf{s}$, represented by $x_{\mathbf{s}}$, is foreground.

(g) $w(\mathbf{s}, \mathbf{s}')$: the value associated with the edge $(\mathbf{s}, \mathbf{s}') \in E$. This value represents the correlation between neighboring superpixels, i.e., the larger the value, more likely $\mathbf{s}$ and $\mathbf{s}'$ will be labelled the same.

(h) **Blackbox**: inference engine provided. Two command line parameters must be provided: γ and `input_file`. $\gamma \geq 0$ and for all edge $(\mathbf{s}, \mathbf{s}') \in E$, the weight used for that edge is $w(\mathbf{s}, \mathbf{s}')^{\gamma}$. `input_file` is the path to the file representing the superpixel adjacency matrix, P_{fg} and weights w. See the provided package for the format of this file and examples.

(i) **Superpixel Segmentation**: ASCII output of the blackbox. This output contains S lines and the ith represents the label of the superpixel $i-1$. The label is either 0 (background) or 1 (foreground).

(j) **Image Segmentation**: Same format as the scribble mask, except that the values for each position (i, j) can be either 0 or 1.

Your job is to implement the items (d), (e), (f) and (g).

Part 1: Implementing your pipeline [40 pts]

a) Describe and motivate your implementation of the pipeline items (d), (f) and (g). [15 pts]

b) Turn in your code. See more details about how to turn in your code in the file `README.txt` in the provided package. [15 pts]

c) Briefly describe how you would implement the blackbox. Don't forget to name the model and algorithm used as well as any transformation necessary. [10 pts]

Part 2: Evaluating your method [25 pts]

For this part of the problem, you will empirically evaluate the effect of γ in your pipeline and the data necessary is in the directory `data/part2` of the provided package. This directory contains 10 subdirectories and each of them contains one image and the following files associated to it: superpixel map, scribble mask and image segmentation.

For $\gamma \in \{0, 2, 4, 6, 8, 10\}$, run your pipeline in each of the images and compare your image segmentation with the one provided. To compare the segmentations, use the provided program `compare_segmentation`. This program takes two arguments, the "ground truth" segmentation, i.e., the one provided by us, and your segmentation. It outputs the score of your segmentation and the lower the better (0 is the perfect score).

Plot your average score of the images versus the value of gamma. What can you conclude from this plot? Discuss the two special cases: $\gamma = 0$ and $\gamma \to +\infty$.

Part 3: Competition [5 pts + 10 extra pts]

Use your pipeline and your favorite value of γ to segment the 10 images in `data/part3`. Report in your write-up the value of γ used and how you chose it. For each image, you should turn in the associated blackbox input file and image segmentation file. The top-3 best average scores will earn 10 extra points.

Problem 2: Social Choice in the Underground [John, 25 pts][2]

Recall the informal proof from class that, given the single peaked preferences of a set of voters on a line, selecting the *midpoint* of the line is not strategyproof, but selecting the *median* of the line is both strategyproof and Condorcet consistent. While this also implies unanimity and disallows dictatorships, we shouldn't worry; this circumvention of the Gibbard-Satterthwaite theorem is due to the severely restricted domain (i.e., single peaked preferences on a line). In this problem, you'll design a social choice function for a slightly less restricted domain, and you'll prove that it is both strategyproof and Condorcet consistent.

A community of gophers has tunneled under the Gates-Hillman Center. Weirdly, the tunnels form a tree $T = (V, E)$, where V is the set of vertices corresponding to tunnel junctions and E is the set of edges corresponding to tunnel segments. Traffic is starting to become a problem, so the gophers bought their first stoplight. They can't decide at which tunnel junction to place the new stoplight, though!

- Each gopher g has a most preferred junction $v_g \in V$.
- Given two vertices u and w, if $d(v_g, u) < d(v_g, w)$, then the gopher prefers u to w. Here, $d(x, y)$ is the sum of the lengths of the edges on the unique path in the tree from vertex x to vertex y.

Design a *strategyproof, Condorcet consistent* social choice function that takes each gopher's preferences and returns a single vertex $v \in V$. You should formally prove both properties about your social choice function.

Problem 3: The Price of Anarchy [John, 20 pts][3]

In class, we discussed routing games in the context of ASes routing traffic on the Internet. In this problem, we'll elaborate on the concept by discussing atomic selfish routing games and the price of anarchy. We define the *price of anarchy* in a game to be the ratio between the cost of the worst (i.e., highest cost) equilibrium flow and that of an optimal flow. For example, in the Internet routing example, if the worst equilibrium had cost 2000, but the best possible routing yielded cost 1500, then the price of anarchy would be $2000/1500 = 4/3$.

We're given the directed graph shown in Figure 3. There are four players $i \in \{1, 2, 3, 4\}$, each with a source vertex marked by s_i and a sink vertex marked by t_i. For example, player 1 has source vertex u and sink vertex v, while player 2 has source vertex u and sink vertex w. Each player can also have a *weight* representing how much traffic she must route (in Figure 3, these weights are not shown are assumed to uniformly be 1, meaning all players route the same amount of traffic).

The cost of sending traffic on an edge is either 0 or x, where x is the amount of traffic being routed along that edge. Players can choose between two strategies, either one- or two-hop. For example, assume both players 1 and 2 both have weight 1. If player 1 takes the two-hop strategy $\langle u, w, v\rangle$ while player 2 takes the one-hop strategy $\langle u, w\rangle$, then player 1 will incur total cost $2 + 1 = 3$ while player 2 will incur total cost 2.

Players are interested in routing traffic from their source vertex to their sink vertex at minimum cost. Players are selfish, so they are only interested in minimizing *their own cost*. This can lead to suboptimal solutions overall.

a) What is the optimal flow for this graph? Define it in terms of each player's strategy, their individual costs, and the overall cost. [4 pts]

b) Is this optimal flow also an equilibrium flow? [4 pts]

c) Find another equilibrium flow. Define it in terms of each player's strategy, their individual costs, and the overall cost. [4 pts]

d) What is the price of anarchy for this instance? [2 pts]

[2] Based on an exercise by Noam Nisan at Hebrew University.
[3] Based on an exercise by Tim Roughgarden at Stanford University.

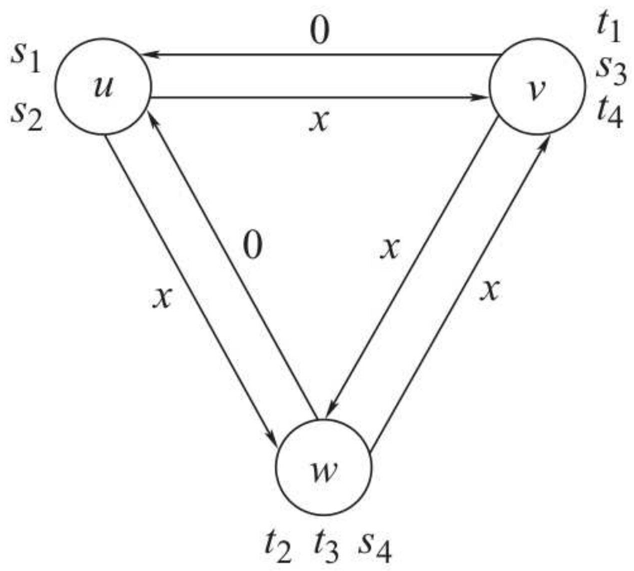

Figure 3: The routing game. Each player $i \in \{1,2,3,4\}$ has a source vertex marked by s_i and a sink vertex marked by t_i. The cost x of an edge is the traffic routed along that edge.

e) In Figure 3, each player's weight is assumed to be 1. This is called an *unweighted* game, because all players route the same amount of traffic. For this problem, modify the players' weights so that the price of anarchy in the resulting weighted instance is exactly $(3+\sqrt{5})/2$. [6 pts]

Problem 4: Setting Prices [John, 25 pts][4]

Two different sellers s_1 and s_2 sell a product to three potential buyers b_1, b_2, and b_3. Each of the buyers wants to buy exactly one unit of this product. The sellers and buyers are connected as follows:

- Buyer b_1 can only buy from seller s_1
- Buyer b_2 can buy from either seller s_1 or s_2
- Buyer b_3 can only buy from seller s_2

Furthermore, each buyer has a budget of exactly 1 dollar. The sellers know this and want to maximize their profits, so they play a *pricing game*, setting a price $p_i \in [0,1]$. Clearly, buyers b_1 and b_3 will be forced to buy from sellers s_1 and s_2 (respectively) at whatever price is named. However, buyer b_2 has more freedom, and chooses the seller who advertises the lower price. If the sellers set the same price (i.e., $p_1 = p_2$), then buyer b_1 will buy from seller s_1.

a) Does a pure strategy Nash equilibrium exist in this game? If so, find one. If not, prove that it does not exist. [15 pts]

b) Do you think a mixed strategy Nash equilibrium exists in this game? You don't need to prove this, just share your intuition. [10 pts]

[4]Based on an exercise by Éva Tardos at Cornell University and Vijay Vazirani at Georgia Tech.