ELEC-E7851 COMPUTATIONAL USER INTERFACE DESIGN, Fall 2021

Aalto University

Antti Oulasvirta

Reminder: All assignments on this course are to be done on an individual basis.

Preparations: Download the notebook of Lecture 2 (a zip file) from MyCourses. Ensure you have numpy and other libraries installed. For this assignment, you do not need to run the Word2Vec part. If you have issues with Anaconda, you can remove the code to a regular Python .py file and do the exercise there. **Note**: In the notebook, PDF images may not be presented in all versions of Anaconda due to a security update that prevents them being displayed. Please take a look at the images in the subfolder. I apologize for any inconvenience.

Submission: Upload two files separately: 1) Your report in PDF format and 2) your notebook in .ipynb format. Note that the .ipynb is not sufficient alone even if you have documented it. We mainly use the PDF for grading and .ipynb for quality assurance.

Deadline is given in MyCourses. Please observe it.

Assignment 2: Computational Design

A2. Understanding objective functions in computational design [0-5 points]

In this task, we experiment with objective functions to learn how they work.

Background: Obtaining a meaningful, effective and efficient objective function is a requirement for computational design applications. An objective function $f(x)$ states, quantitatively, what makes a design good. More technically, it is a *function* that assigns an *objective score (or objective value)* to a *design candidate*. The design candidate that obtains the highest (or lowest, when minimizing) score is the *optimum design* (recall the difference between a global and local optimum). An objective function can represent anything that end-users regard desirable in interaction. It is often construed by reference to surface features of the design (e.g., items associated with each other should be close to each other; coloring should be similar to other designs in this domain). The downside of this approach is that such features are just proxies for what we really want to optimize for, which should relate to end-users. We can also construe models by reference to end-users, such as their expected performance or experience (e.g., 'task should be completed as quickly as possible'). There are five principled ways to obtain an objective function:

- A predictive model based on theory;
- Heuristics based on the modeler's own domain knowledge;
- Standards, guidelines, and design patterns expessed heuristically;
- Predictive models based on supervised learning: using machine learning to obtain a model that associates candidate designs to estimates of value to end-users;
- Human-in-the-loop learning: learning a model by experimenting on end-users (e.g., Bayesian optimization).

Your goal is to formulate objective functions for improving the generated menu systems. We provide Fitts' law -based (obj_fitts), association-based grouping (obj_assoc), and the foraging-based functions as the starting point.

Preparations: In the notebook, launch the cell showing the interactive menu and click commands to produce some click data (data.csv). Then run the optimization cell below using obj_fitts as the objective function. It should produce menu designs that are ordered by decreasing frequency according to Fitts' law. This was demonstrated during the lecture.

Four sub-tasks:

1. **Comparing two objective functions**: Try the foraging-based objective function and compare it to the weighted sum approach (obj_assoc and obj_fitts). How do the results differ and, more importantly, why? Tip: Be mindful of the effect of *weight* in the multi-objective case. (1 point)
2. **Creating a new objective function for users' expectations:** Let's assume that users have learned to expect certain items to in certain locations. **Your task**: Develop a new objective function (called obj_expectation) that ensures that items are in expected locations. The function is given as input the expected location of each item (or those that are associated with expectations) and the strength of that expectation. Show the results in a single-objective setting, optimizing for that objective only. Then show results for the multi-objective setting where you have e.g., obj_fitts or obj_assoc as the other objective. **Tip**: The challenge is to formulate this as a function that assigns a real value to a candidate design. If you implement this as a *constraint*, it will be too harsh and prevent moving items from expected locations. (1 point)
3. **Modifying obj_assoc: Group anchors**: If you look at the results that obj_assoc produces, it can happen that the most descriptive items of a group sits in the *middle* of the group. A user might thus have to read some undescriptive items before seeing this "anchor". **Your task**: Extend obj_assoc to place the most descriptive item of a group as its *anchor*. An anchor is an item that sits at the *top* of the group. It is supposedly the most descriptive item and should therefore be first. Define a function that captures this intuition and show that it can work such that it does not distract the other objectives. (2 points)
4. **Sensitivity analysis** refers to the study of how *sensitive* an optimizer is to *changes* in its inputs. Your task here is to systematically *perturb* a task instance. In practice: you should systematically change frequencies, associations, and/or menus, and report how sensitive the results are to these perturbations. You can pick which objective function you use (1 point).

PDF report: For subtask #1: Present screenshots of outputs and discuss their pros and cons. For the rest: 1) present your approach verbally, 2) then show pseudocode or screenshot of real code and explain it, then 3) present results with screenshots, and 4) finally draw conclusions and, if relevant, discuss the pros and cons of your approach.