Homework 8

[ 100 points - due by 11:59 pm, Sunday, April 9, 2017 ] *("next" Sunday relative to writing this…)*

Submit these files to the CS submission system at the usual place by 11:59. You may work on your own or with 1-2 partners on the programming portions of this assignment. (The reading/response is individual only.) Groups larger than 3, please split into smaller groups! Remember that partners need to work in the same physical location, share composition time equally (or each compose on their own machines) and be fully equal owners and producers of their work. *Have fun treeing (and foresting)!*  [cs35 homepage](https://www.cs.hmc.edu/~dodds/cs35/)

**Downloads**

There's one (zipped) starter file to download -- grab it at the start of class & follow along:

* [The zip file, hw8.zip, to start this week's problems…](https://drive.google.com/open?id=0BwPWh-3AmiLxOHVSZFB6YXQ4a0U)

**Submission**

**Overview** Please submit an archive named **hw8.zip** with these files:

**hw8pr1.py** [40 pts] pixels, images, and the k-means algorithm… [no lab problem this week]

**hw8pr2.py** [??? pts] interpreting pixel data with neural networks

**project\_plan.txt or .doc** [10 pts] project proposal and initial plan

**Project\_progress.py or .txt or .doc** [same] (or any filename) your progress in this first week...

As usual, submit your reading response in its own spot at the [submission site](http://cicero.cs.hmc.edu/).

As always, extra-credit is available for posting code and a write-up of any one of these problems to your GitHub repository (be sure to let us know you've done this -- and provide a direct link)

**Reading** (No reading problem this week -- but, likely, research on possible projects and libraries/material that will help.)

**[Part 1] Problem 1: k-means and image *posterizing*...**

[40 pts; pixels and the k-means algorithm: *posterizing!*]

* This problem asks you to run/alter the code in the **hw8pr1.py** file.
* This week presented an algorithm called *k-means,* which seeks to find the **k** datapoints that "best" represent a certain dataset, where k is a parameter chosen by the user. For instance, with k==2, k-means computes the *two* datapoints that best capture the dataset. With k==3, k-means computes the *three* values that best capture the dataset, and so on.
* Starting from the code examples in hw8pr1.py, choose at least two of your own images and run the k-means algorithm on them.
* **Part A** For each one of the two images you choose, run k-means with at least four different values of k. Also, display the original images and the "color summaries" (horizontal bars) for your two images.
  + For each of the two images, include a screenshot (or saved image) of the four "color summaries" -- together in a single submitted image.
* **Part B, Choice 1** Next, you can choose one of two paths to finish this problem - the "pixel/posterizing" path or the "algorithmic" path:
* **the posterizing path** This path asks you to create several new images for each of your two (or more) originals. The new images should be simplified in their color content.
  + Specifically, you should write a function (with helper functions, as needed) that *replaces the pixels in the original image* with the mean (from k-means) each pixel is closest to. This will yield a [*posterized*](https://en.wikipedia.org/wiki/Posterization) image with only k different pixel-colors total than the original image. To accomplish this, you will need to
    1. loop over all of the pixels in the original image
    2. for each one, find the mean (from k-means) it's closest to. Use the usual "Pythagorean-style" distance here
    3. replace the pixel with that closest mean
    4. render the image in a new plot
* ***Extra cool*** points (and EC) for creating "boundaries" between the k different colors - you choose what color the boundaries should be: black, grayscale, white, etc. This can be done by replacing each pixel that is NOT completely surrounded by its own color with one of the boundary pixels
* Whether or not you create the boundaries, choose at least two pictures and values of k -- tuned for them specifically -- for which you like the resulting "posterized" artistic effect. Include the posterized images
* Also, include a note at the bottom of hw8pr1.py describing briefly how this problem went, and how you chose the values of k for your two images.
* **Part B, Choice 2: the algs path** This option asks you, instead, to build the k-means algorithm yourself - and verify it by using it on the images from the first part of this problem. To accomplish this, you will need to
  1. [make sure you understand the k-means algorithm](http://www.onmyphd.com/?p=k-means.clustering&ckattempt=1#h2_algorithm)
  2. [here is Wikipedia's too-concise take](https://en.wikipedia.org/wiki/K-means_clustering#Standard_algorithm)
  3. and we'll restate it here: ***first***, initialize k centers at random
  4. ***next***, determine which center each data point belongs to (is closest to)
  5. ***then***, re-compute the locations of the centers as the means (averages) of those data points belonging to it
  6. ***continue*** until convergence
* Be sure to test your k-means algorithm on small data sets you create by hand -- not images/pixels. Make sure there are clear clusters and a clearly correct value of k.
* Once you know it works for that, try it on one or more of the images from the first part of this problem.
* When it works - and it will - you'll have implemented your own k-means!
* Either way, you should include a comment at the bottom of hw8pr1.py with a reflection and explanation on the k-means path you tried -- as well as the results (your own image, with its histogram of pixels or your own data, distilled into k-means by your alg.)
* ***EC Options***: Since there are two paths to finishing this problem, you're invited to complete ***the other path*** for EC of up to +20pts.

Didn't know if you wanted this problem in week 7 or week 8

**Problem 2: Neural Networks with Pixel Processing**

[??? points]

* This problem asks you to run/alter the code in the **hw8pr2.py** file.
* In this problem, you will train a neural network to recognize pictures of letters of the alphabet!
* Download the necessary files [here](https://github.com/ScriptingBeyondCS/CS-35/tree/master/week_8).
* Open up **letters.jpg**. In each row are the capital letters of the alphabet, each in 50 different fonts. Each cell is 10 x 10 pixels.

A similar example with the handwritten digit set digits.png (from hw4) is performed at [opencv.org](http://docs.opencv.org/3.2.0/d8/d4b/tutorial_py_knn_opencv.html) (code also provided in cv\_digits.py). Use the code under "OCR of Handwritten Digits" as a template as you construct your own code.

Note: The digits data differs within its columns, while the alphabet differs within its rows. Keep this in mind as you assemble your data.

**TO DO:**

Assemble Pixel Data

* Grab letters.jpg using the cv2 library and convert it to grayscale.
* Divide the picture into individual cells and convert it into an array.
* Write lines to randomly shuffle the rows before you divide them into training and testing (the method is [np.random.permutation](https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.permutation.html), in case you've forgotten).
* Divide the data into 38 training rows and 12 testing rows. Reshape each of them so that you have 988 (38\*26) or 312 (12\*26) rows (representing each individual letter) and 100 columns (the brightness value of each grayscale pixel in the 10 x 10 cell).
* Now create labels for the training and testing data (this is mostly done for you).

Machine Learning

* Now train a neural network to predict the testing labels (refer to hw5—boston.py for a refresher on NNs)
* Compare the NN's predictions with the test labels. Compute your accuracy on your testing data.
* Use cross-validation to find the best parameters for your NN (layer sizes, # of hidden layers) and try to get an accuracy of over 85%! (best so far is 90%)

**[Part 2] Final project: proposal/plan and getting started...**

[10 pts; a one-page outline of a project proposal and an initial plan]

* Submit your project idea and plan in **project\_plan.txt** or .doc or .pdf or ...
* [**Overall goals/tasks**] *[ include this in your hw8.zip archive ]*

For the final project, we ask each cs35er to propose and pursue a Python project of their own design. The goal is to create a medium-sized project that yields insight into something of interest to each group.

* This can be learning more about a specific library and its capabilities through a demonstration and/or
* It can emphasize *using* Python libraries to tackle a problem of interest...

***Features*** Here are a list of features we hope each project will use/have.

* **An exploratory task and motivation**
  + Something you're interested in, professionally or personally, either long-term or "just now" is welcome
* **One or more Python libraries** whose capabilities will be a central part of the project
  + The libraries you used through cs35 are the most natural starting point
  + However, you can choose something entirely different, if you wish
  + Do look for overlaps, if they exist - I'll mention some ideas this week
* **One or more datasets**
  + For this project, datasets are broadly defined, e.g., images, sounds, an NLP corpus, as well as the kinds of data traditionally considered "datasets"
  + Even if your project is primarily algorithmic, there will still be testing...
* **A place or places to start** 
  + documentation, tutorial, and/or examples, e.g., online
  + For new libraries, I encourage you to check out how accessible its documentation is -- and whether it has a tutorial available. A path that was often successful in the past has been to start by working through a tutorial and then continue by customizing the library to your own tasks/problems...
* **A plan for going *beyond* the structure already provided** 
  + Thus, I ask you to be very deliberate about where you branch off from a guide or tutorial so that you make sure you create something of your own. It does not need to be entirely new -- it's that the creative portion needs to be thoughtful and novel (for you)
  + The last part of this homework is to make some progress in this initial week.

If you have in mind something that doesn't quite fit into the general breakdown above, definitely chat with me about it!

***Project proposal/plan*** Write up a 1 page document (txt or pdf or docx are best). More is ok, but not required. Your proposal + plan should address each of the facets above:

* Overview of your idea and big-picture goals for the project
  + a description (~1 paragraph) of - overall - what you'd like the project to do/investigate - including a bit of the motivation behind it
  + Especially encouraged: overlaps with other projects/interests you have in your major, other academic work, or something entirely non-academic
* Specific Python library/libraries
  + one (or more) Python libraries/resources you'd like to investigate/use
  + Since it's cs35, yours should be largely a *Python-based* project
    - it can certainly overlap with other languages
    - it can also be to learn a new Python library or framework
    - it's ideal to have an online tutorial to get you started
    - as long as you go ***beyond*** that tutorial
* Online resources from which you will get started: a great template:
  + work through a library's online or other tutorial/examples
  + then, do something of your own that extends those examples
  + Be sure to explain the novel vs. the provided portions of your project
* Proposed timeline for progress/subgoals
  + Propose a short timeline that's concrete as possible -- at least for the first few tasks…
    - What you'll work on this week
    - What you'll seek to have accomplished by your presentation
    - What you'll want to have completed by the end of the project
  + As we progress, you'll be able to adapt -- to align with what's interesting and feasible! Still, it's still good to have a starting list of tasks.
  + Here's our timeline for the final project and presentations:

***Timeline*** Here is a summary timeline of cs35's project. Note that one of the key deliverables is a presentation to the cs35 class during the final two weeks of our meetings:

* **due 4/9**: your project proposal (this assignment) and "getting started"
* **On Mon. 4/17 and Mon. 4/24**:
  + Presentations by 9-10 teams per class, 8-10 minutes per team
  + Present your idea, your progress thus far, ideally with a demo + what's still to go
    - *More detail and examples on presenting next week*
  + We'll invite feedback from the rest of the class (we'll scaffold it)
  + We will also ask for each team to propose a transition from one talk to the next
  + **Goal**: ideas, opportunities, and insights about resources out there
* **For the day you're presenting 4/17 or 4/24**:
  + The only deliverable that week is your presentation materials.
  + In addition, I ask you to meet with me sometime in the week beforehand, to go over presentation plans and project progress. This is a brief meeting (15-20 minutes).
* **For the day you're NOT presenting 4/17 or 4/24**:
  + Submit a zip file with
  + [1] a one-page update of your progress up to that point
  + [2] the software/other resources you've created and explored up to that point
* **Class doesn't meet in May, but the final projects are due**
* **For graduating seniors, due on 5/5**: final project deliverable
* **For everyone else, due on 5/13**: final project deliverable
  + Report and reflection (sections to be detailed next week)
  + Software (with docstrings!)
  + Results (files, images, or whatever the results might be…)

***Other Details***

* Scope: the project should be about a 4-week scope
  + that is, effort of about 4-weeks of the usual Python hwks.
* Chat with us - and stay adapatable - about keeping the scope the right size
  + not too small: some open-ended challenge/creativity must be present
  + not too big: easy to become "solve all problems"
  + it's almost always possible to identify *parts* of a bigger problem to solve
* Grading: More detail next week. Roughly worth four hw-assignments:
  + 1/2 for the final deliverable and report/reflection
  + 1/3 for the project presentation
  + 1/6 for the proposal, student feedback, and intermediate updates

**[Part 3] Problem 3: Getting started!**

[the same 10 pts as Problem 2 -- this week's hw will be out of 50 points, not 100]

* since many people were still thinking about things at the end of the week...
* This problem asks you to start making progress on your project...

Along with the project-proposal and plan, above, I ask you to spend some time this week with the initial tasks of exploring your project. You should include a ***one-page*** (or so) description of your progress in this first week. For example,

* [**desirable**] Did you find one or more libraries/resources… what are they?
* [**desirable**] Have you successfully installed/run the software… ? How did it go?
* [**desirable**] What is your plan for getting to know the library/resources?
  + Is there a tutorial? A set of examples? Other online (or offline) guides?
  + What are they? How far have you made it in this first week?
* [**wonderful, but not required**] Have you been able to adapt one or more of the library's resources to something related to your own piece of the project
* And, it's open-ended from there…

In essence, the goal here is to (a) make sure the software and other resources are working by the end of the first week and (b) you've successfully started using it -- and have a concrete plan for what to pursue next.

It's encouraged to include any code you've written so far -- along with a comment on how it went, that is a great item to hand in as this "getting started" assignment. Alternatively, you're welcome to create a separate page describing progress so far.

(The exact name of the file is not important - but please include "progress" somewhere in the name, so we know! :-)

*Good luck diving into your projects, everyone!*

And, as always, the ...

**Eternal Extra-credit: Showing off your results…**

[up to +5 pts extra-credit...]

* As with each week, you're invited to include both your source code and a short write-up of one of the week's problems within your GitHub repo(s). Images and other visuals, of course, are welcome. If you do this, let us know (and provide a direct link :-)