Image Classification

This model would utilize a classifier, such as k-nearest neighbor. The data would be classified according to the object depicted in each image, and thus the samples would need to be labeled manually. The test data would then consist of other images depicting the same objects – stripped of labels, of course. In pre-processing, the images could be divided into a fixed number of discrete pixel grids, which would comprise the data features. The model would likely benefit from a feature reduction procedure, as the images are bound to include a moderate-to-significant degree of noise in the form of image regions that do not depict the object in question.

The benefit of this project is the abundance of available data. One possible approach would involve harvesting images from Google reCAPTCHA forms, which could be automated to some extent. Since the reCAPTCHA images are designed specifically to fool this sort of model, it might be necessary to derive data from simple Google Image searches for objects such as cars, stairs, and fire hydrants to train and test a model with satisfactory performance. Nonetheless, testing the model's performance on reCAPTCHA images would be an interesting experiment. In any case, the internet has no shortage of generic images, so data procurement would not present much of a challenge.

A reliable method for breaking reCAPTCHA is described in a white paper[1] I read a while back. If the model were tailored to reCAPTCHA images, some of the pre-processing tricks described in this paper could be utilized to increase accuracy. Various image classification libraries with Python support already exist for this sort of task, but since the point of the project is to use low-level machine learning algorithms themselves, I assume we will not be able to use these libraries. As such, it might be difficult to create a high-performance model for this task.

Facial Recognition/Comparison

Subject to the same limitations as the above idea, this idea might also prove difficult to implement. This model would take a similar approach to the one described above, only the data would be comprised of faces – most likely faces of celebrities taken from Google Images. Like the model described above, this model could benefit from feature reduction prior to classification. Another interesting approach would involve creating clusters of "similar" faces. A test sample could then be added to a cluster, and the most common label in the cluster could be applied to that test label. The result would indicate which face in the training set is most similar to the face depicted by the test sample. Alternatively, the percentage of samples of each class in the cluster could be taken as an indication of similarity to each face in the training set.

Without advanced image processing utilities, implementing this model would likely prove infeasible. If we were able to use image processing utilities for pre-processing, however, we might be able to isolate facial regions that would make better features for classification and/or

[1] https://www.blackhat.com/docs/asia-16/materials/asia-16-Sivakorn-Im-Not-a-Human-Breaking-the-Google-reCAPTCHA-wp.pdf

clustering. Otherwise, we would have to resort to dividing the images into pixel grids, which would probably be insufficient for this particular task. With image processing utilities, we could also automate data collection and labeling. This would involve harvesting Google Image search results for a particular name, isolating the facial region of each image, and exporting this region to an image file that would serve as a data sample. Documentation of facial recognition systems is widely available online, and these resources could aid in the construction of our model.

Election Data Analysis

This would be another classification problem. Any election could be used for this project, but the most intuitive one, and the one with the most abundant data and most clearly defined trends, would likely be the 2016 presidential election. The model's objective would be to predict which presidential candidate any particular county/district in the U.S. voted for in the 2016 presidential election based on demographic information, such as age distribution, income distribution, and racial/ethnic composition. Of course, various mitigating factors exist such as percentage of registered voters and voter turnout, but the model would assume that every citizen of every county voted in the election.

Datasets documenting county-by-county vote share can be found online, and county demographic information is available on the census website. With some web scraping and CSV processing, the collection and labeling of data could be automated. Each county would be represented as a set of demographics features, and its label would be the candidate for which it voted in the 2016 presidential election. Since some of these features are likely better indicators of candidate preference than others, the model could benefit from feature reduction, and feature reduction would also yield an interesting analysis of candidate preference by demographic group. Our first task would be to choose a classifier that works well with continuous values and many features. From there, classification would proceed as usual. Of all three project ideas, this project is likely the most viable, assuming we are able to determine which classifier can best accomplish this task.