CS 441

HW 4: Dealing with Data

Due Date: April 17, 2023 11:59pm

In this assignment, we explore clustering, retrieval, density estimation, and data visualization using our old friend MNIST digits, as well as a few images. The aims of this homework are:

1. To experiment with kmeans and its ability to group data in useful ways without supervision
2. To see how fast search techniques can make nearest neighbor retrieval much more efficient
3. To compare different strategies for estimating the probability density functions of variables
4. To learn how to use data projections, particularly PCA, to compress and visualize

Read this document and the report template and tips and tricks before beginning to code.

- Report template
- Tips and Tricks
- Starter code

1. Clustering and Fast Retrieval [40 points]

Efficient querying of high-dimensional data is important for many applications, including retrieval, clustering, and K nearest-neighbor classification. In this part, we will use the Faiss (Facebook AI Similarity Search) library to explore kmeans and KNN on the MNIST data.

K-means: Use faiss to "train" Kmeans (K=10) on `x_train`. Then, get the cluster indices `cluster_idx` for the samples in `x_test`. Display the average sample distance to its centroid, purity, and centroids with the following code.

```
print(dist.mean())

purity, counts = get_purity(y_test, cluster_idx)

idx = get_cluster_order(counts)

display_mnist(kmeans.centroids[idx, :],1,K)
```

Repeat this for K=10, 20, 30, 40, 50, … 100 and display the cluster center visualization and record mean distance and purity for each. You can experiment with `niter` and `nredo` (see documentation). Include line plots of K vs. purity and K vs. mean distance in your report. Also include the displays of centroids for K=10,20,30.

Also, answer a few questions in your report:

1. As you increase K, do you expect the purity to increase? Why or why not?

2. In a given run, is the average distance of a sample to centroid guaranteed to monotonically decrease (or not change) with each iteration? Why or why not?
3. If you do enough iterations, is Kmeans guaranteed to give you the optimal clustering that minimizes the sum of distances between each sample and its center? Why or why not?
4. Does improving the Kmeans objective (i.e. achieving lower mean squared error) necessarily improve expected purity? Why or why not?

1-NN Revisited: Remember how this took tens of minutes for HW 1? Use the library for more efficient nearest neighbor classification. Don't worry, it will be much faster. You can try two methods, initialized by

```
index_lsh = faiss.IndexLSH(dim, nbits) # LSH

index_flatl2 = faiss.IndexFlatL2(dim)  # Brute Force
```

In either case, you add the data to search for with `index.add(X_data)` and perform the search with `index.search(X_query,1)`. Try using both methods for 1-NN using `x_train` and `x_test`. Try varying `nbits` to see how it affects time and accuracy (e.g. `ndim/2`, `ndim`, `ndim*2`). If your accuracy is roughly 10% (chance performance), check the shapes of `y_test` and the returned indices to make sure they are the same. For each method, report test error and the time to add and to search, using `time.time()`. See documentation of the different methods available, and feel free to try out more.

2. Estimating PDFs [35 points]

We will experiment with three methods of estimating probability density functions (PDFs). In this case, given an RGB image and a box containing part or all of an object of interest, we will estimate the probability of observing a particular color inside the box, compared to the probability of the image as a whole. If rgb_i is the color of a particular pixel, we can write

$$score = log[P(rgb_i|box)/P(rgb_i|image)]$$

as the log ratio probability of observing a color given that it came from inside the box versus randomly drawn from the image as a whole. This gives us a good pixelwise score for segmenting out the object inside the box. To keep it simple, we will compute the box color density using a cropped image (which prevents having to record and use the box coordinates). We've included one example from Flickr.

Estimate the color PDF of the crop image and the full image using four methods:

1. Estimate the probability of each color channel separately using discrete pdfs (by counting) and then model the joint probability as the product of per-channel probabilities. Try anywhere from 2 to 256 bins per channel.
2. Estimate the joint probability by clustering the pixel colors in the whole image and then estimating the probability of each cluster. Try varying the number of clusters , e.g. between 16 and 512.

3. Estimate the joint probability using a Gaussian mixture model. Try varying the number of Gaussian components (e.g. between 1 and 10) and whether using full covariance or diagonal covariance.

For each method, display the image, the score map, thresholded score map, and a thresholded image with some threshold of your choice (typically in the range of -2 to 2). Helper code is provided for these displays. Also report your parameters (e.g. number of bins, K, number of Gaussian components, covariance matrix type). Each method can provide good (but not perfect) segmentations.

3. PCA and Data Compression [25 points]

On MNIST. Compute the principal components using PCA.

1. Display the first 10 principal components using the same tool that is used to display centroids
2. Scatterplot `x_train[train_indices['s']]` for the first two PCA dimensions
3. Plot cumulative explained variance (`explained_variance_ratio_`)
4. Select the smallest number M of principal components that explains at least 90% of variance. Then, compress the data using the first M principal components, and compute the total time and test error for 1-NN using the brute force Faiss method.

How does the time and performance compare to 1-NN without compression using brute force or LSH?

4. Stretch Goals

1. For Part 2.1, use PCA to rotate your data so that the axes lie along the principal directions and then use the per-channel probability estimates. Compare score maps and segmentations to the original approach for Part 2.1. [15 pts]
2. For Part 2, try the same three methods on an image of your own choosing. Pick an image and rescale to somewhere between 512 and 600 on the larger side. Then save a cropped version that contains the object of interest. Again, for each method, display the image, score map, and thresholded image. [10 pts]
3. For Part 3, plot the points of `x_train[train_indices['s']]` using at least two of t-SNE, MDS, and Linear Discriminant Analysis, and display these plots alongside the PCA scatterplot. [15 pts]

Submission Instructions

Submit three files: (1) completed report template; (2) an html or pdf version of your Jupyter notebook; and (3) a zip or ipynb of your notebook and any other python files needed to run your code. Be sure to include your name, number of points claimed, and acknowledgments. **The**

report is your primary deliverable. The notebook and code are included for verification/clarification by the grader.