CSC320: Assignment $\#\sqrt{-1}$

Due on Friday, January 1, 2016

Firstname Lastname

January 24, 2015

$Dataset \ description$

The dataset consists of 2359 grayscale 25×25 -pixel of images of the letter "a," rendered using various fonts. The letter appears in both uppercase and lowercase, and there is considerable variation in the appearance of the letters. However, examining the dataset, it appears that there are more print letters than letters in other fonts, and most of the letters are lowercase. A random sample of 25 "a"s is shown in Figure 1.



Figure 1: A random selection of 25 "a"s from the dataset. Generated using display_25_rand_images() (line 72)

Advice: when describing a dataset, give a general description, and mention the things that would be important for a person who is working on the same problem that you are working on.

$Applying \ an \ algorithm: \ initial \ exploration.$

Using the given dataset of "a"'s, Figure 2 displays the first 25 principal components and Figure 3 displays a the mean image in the dataset. For these results to make sense, look back at Figure 1, which shows a sample of images from our dataset. These images were generated using the function **display_save_25_comps** (lines 86-94 and 154-155 of pca_example.py).



Figure 2: The first 25 principal components of the "a"s. Generated using lines 154-155 and display_save_25_comps().



Figure 3: The mean image. Generated using line 154.

Interestingly, the mean image and the first principal component are extremely similar. Adding a scalar multiple of the first principal component would simply make the mean image darker or brighter. The principal components seem to be of the form "emphasize one type of 'a' while subtract another kind of 'a'." *Advice: make observations about the output, and try to explain them.*

Reconstructing noiseless image using varying numbers of principal components.

We reconstruct the noiseless image 1000_t.jpg using the first N principal components, for various N. The quality of the reconstruction increases as n increases. For just 1 principal component, we simply get the mean image. For 25 principal components, we are basically able to reconstruct the letter. The output for 100 principal components is fairly close to perfect. The reconstruction is perfect when we use all 625 principal components, as expected. (Note: $625 = 25 \times 25$ is the dimensionality of the images.)

Advice: describe the quality of the results that you get. To the extent possible, explain why things are the way they are (e.g., why the reconstruction is perfect for 625 components.



(a) Reconstruction using1 principal component.



(d) Reconstruction using 100 principal components.



(g) Reconstruction using 400 principal components.



(b) Reconstruction using5 principal components.



(e) Reconstruction using150 principal components.



(h) Reconstruction using all 625 principal components.

Figure 4

Figure 4 demonstrates the effects of increasing the number of principal components. These images were produced using the function **get_reconstruction** (lines 65-70 and 158-164 of pca_example.py).



(c) Reconstruction using 25 principal components.



(f) Reconstruction using 200 principal components.

Reconstructing images where salt-and-pepper noise was applied

Noisy images are created using the function salt_and_pepper_noise() (lines 98-106 of pcalexample.py). The function salt_and_pepper_noise() randomly sets a subselection of pixels to 0 or 1, while *noise_prop* is the proportion of the pixels set to either 0 or 1. Once the salt-and-pepper noise is applied, the images are reconstructed using 200 principal components using the function get_reconstruction() (lines 65-70 and 167-317 of pcalexample.py).

We vary the proportion of pixels that are perturbed using salt-and-pepper-noise.

Some results are shown in Figure 5, Figure 6, Figure 7, Figure 8.

For moderate amounts of noise, we are able to get a reconstruction that looks similar to the original image, without the noise.

After the proportion of pixels that are perturbed becomes greater than a certain threshold, (the threshold varies from one letter to another), the reconstruction gets worse and worse and eventually stops producing an image that looks like an 'a'. The threshold in Figure 5 and in Figure 7 is around 60% (*noise_prop* = 0.6). That of Figure 6 is around 45% (*noise_prop* = 0.45). Figure 8 has the lowest threshold out of these four examples, at around 20% (*noise_prop* = 0.2).

Advice: look at the results, point out where the algorithm works, and where the algorithm doesn't work.



(a) Original image



(b) Image 5a with 5% noise



(d) Image 5a with 10% noise



(f) Image 5a with 20% noise



(h) Image 5a with 30% noise



(j) Image 5a with 45% noise



(l) Image 5a with 60% noise



(c) Reconstruction of Image 5b using 200 principal components



(e) Reconstruction of Image 5d using 200 principal components



(g) Reconstruction of Image 5f using 200 principal components



(i) Reconstruction of Image 5h using 200 principal components



(k) Reconstruction of Image 5j using 200 principal components



(m) Reconstruction of Image 5l using 200 principal components



(a) Original image



(b) Image 6a with 5% noise



(d) Image 6a with 10% noise



(f) Image 6a with 20% noise



(h) Image 6a with 30% noise



(j) Image 6a with 45% noise



(l) Image 6a with 60% noise



(c) Reconstruction of Image 6b using 200 principal components



(e) Reconstruction of Image 6d using 200 principal components



(g) Reconstruction of Image 6f using 200 principal components



(i) Reconstruction of Image 6h using 200 principal components



(k) Reconstruction of Image 6j using 200 principal components



(m) Reconstruction of Image 6l using 200 principal components



(a) Original image



(b) Image 7a with 5% noise



(d) Image 7a with 10% noise



(f) Image 7a with 20% noise



(h) Image 7a with 30% noise



(j) Image 7a with 45% noise



(l) Image 7a with 60% noise



(c) Reconstruction of Image 7b using 200 principal components



(e) Reconstruction of Image 7d using 200 principal components



(g) Reconstruction of Image 7f using 200 principal components



(i) Reconstruction of Image 7h using 200 principal components



(k) Reconstruction of Image 7j using 200 principal components



(m) Reconstruction of Image 7l using 200 principal components



(a) Original image



(b) Image 8a with 5% noise



(d) Image 8a with 10% noise



(f) Image 8a with 20% noise



(h) Image 8a with 30% noise



(j) Image 8a with 45% noise



(l) Image 8a with 60% noise



(c) Reconstruction of Image 8b using 200 principal components



(e) Reconstruction of Image 8d using 200 principal components



(g) Reconstruction of Image 8f using 200 principal components



(i) Reconstruction of Image 8h using 200 principal components



(k) Reconstruction of Image 8j using 200 principal components $% \left({{{\bf{k}}_{\rm{B}}}} \right)$



(m) Reconstruction of Image 8l using 200 principal components