Yohann Smadja
smadja@post.harvard.edu
CS181-S16

Assignment #4
Due: 5:00pm Feb 1, 2016

Collaborators:

Homework 4: Clustering

There is a mathematical component and a programming component to this homework. Please submit ONLY your PDF to Canvas, and push all of your work to your Github repository. If a question requires you to make any plots, please include those in the writeup.

Problem 1 (The Curse of Dimensionality, 4pts)
In d dimensions, consider a hypersphere of unit radius, centered at zero, which is inscribed in a hypercube, also centered at zero, with edges of length two. What fraction of the hypercube's volume is contained within the hypersphere? Write this as a function of d. What happens when d becomes large?

Solution

volume hypersphere = $V_s = \frac{\pi^{d/2}.radius^d}{\Gamma(d/2+1)}$

volume hypercube = $V_c = length^d = 2^d$

with $radius = 1$ and $length = 2$

$\frac{V_s}{V_c} = \frac{\pi^{d/2}}{2^d.\Gamma(d/2+1)}$

We can see that the limit of this ratio goes to 0 when $d \to +\infty$. The volume of the hypercube goes to infinity while the hypersphere converges to 0.

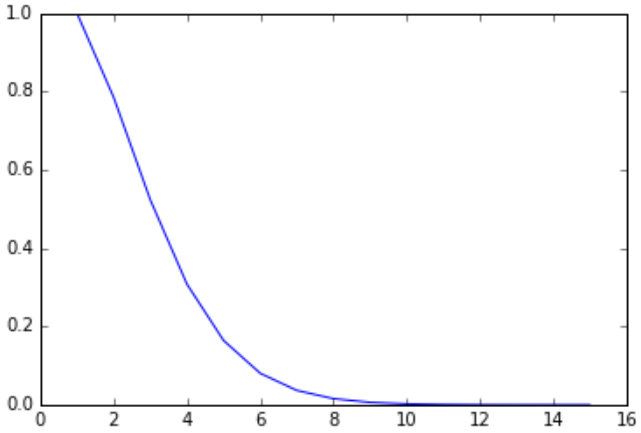


Figure 1: Ratio of the volumes of unit hypersphere and hypercube of length 2 up to the dimension 15

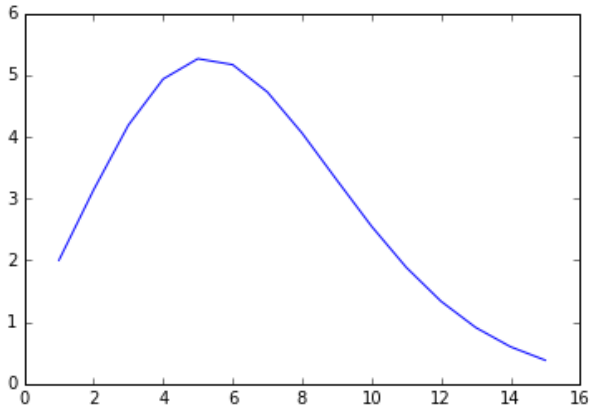


Figure 2: Volume of the unit hypersphere up to the dimension 15

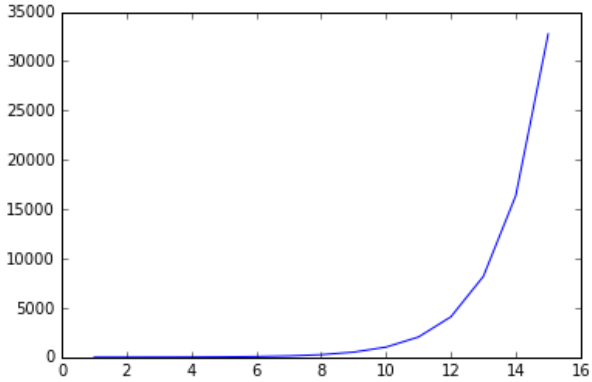
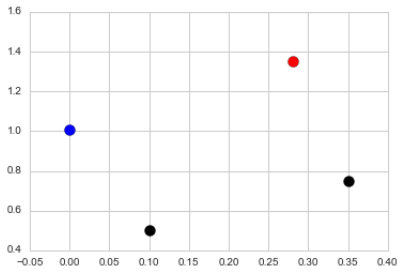


Figure 3: Volume of the hypercube of length 2 up to the dimension 15

Problem 2 (Norms, Distances, and Hierarchical Clustering, 5 pts)

Consider the following four data points, belonging to three clusters: the black cluster ($(x_1, y_1) = (0.1, 0.5)$ and $(x_2, y_2) = (0.35, 0.75)$), the red cluster $(x_3, y_3) = (0.28, 1.35)$ cluster, and the blue cluster $(x_4, y_4) = (0, 1.01)$.



At each step of hierarchical clustering, the two most similar (or least dissimilar) clusters are merged together. This step is repeated until there is one single group. Different distances can be used to measure group dissimilarity. Recall the definition of the l_1, l_2, and l_∞ norm:

- For $\mathbf{x} \in \mathbb{R}^n$, $\|\mathbf{x}\|_1 = \sum_{i=1}^n |x_i|$
- For $\mathbf{x} \in \mathbb{R}^n$, $\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^n x_i^2}$
- For $\mathbf{x} \in \mathbb{R}^n$, $\|\mathbf{x}\|_\infty = \max_{i=1}^n |x_i|$

Also recall the definition of single-link distance, complete-link distance, and average-link distance between two clusters:

- Single-link clustering: for clusters G and H, $d_S(G, H) = \min_{i \in G, j \in H} d(i, j)$
- Complete-link clustering: for clusters G and H, $d_C(G, H) = \max_{i \in G, j \in H} d(i, j)$
- Average-link clustering: for clusters G and H, $d_A(G, H) = \frac{1}{|G||H|} \sum_{i \in G} \sum_{j \in H} d(i, j)$

Warm up question. Draw the 2D unit sphere for each norm, defined as $\mathcal{S} = \{x \in \mathbb{R}^2 : \|x\| = 1\}$. Feel free to do it by hand, take a picture and include it in your pdf.

Main question. For each norm (l_1, l_2, l_∞) and each clustering method (single, complete, or average link clustering), specify which 2 clusters would be the first to merge.

Solution

1 - We start by drawing 3 unit spheres with the 3 different norms l_1, l_2, l_∞. The norm l_1 has the shape of a diamond, the l_2 sphere is a circle and finally, the sphere of l_∞ has the shape of a square.

2- Below is the plot of the points using the same scale for both the x-axis and the y-axis. It looks a little bit different than on the chart above.

We denote:

z1=[0.1,0.5] #black

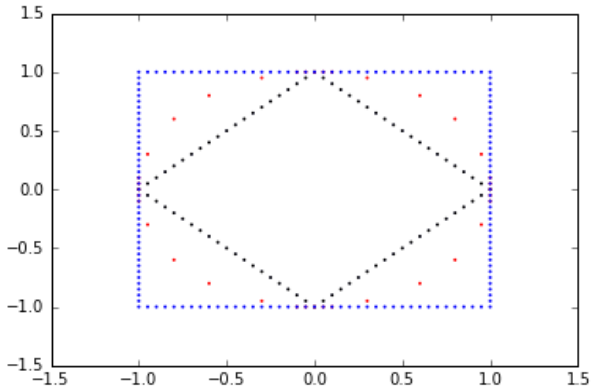
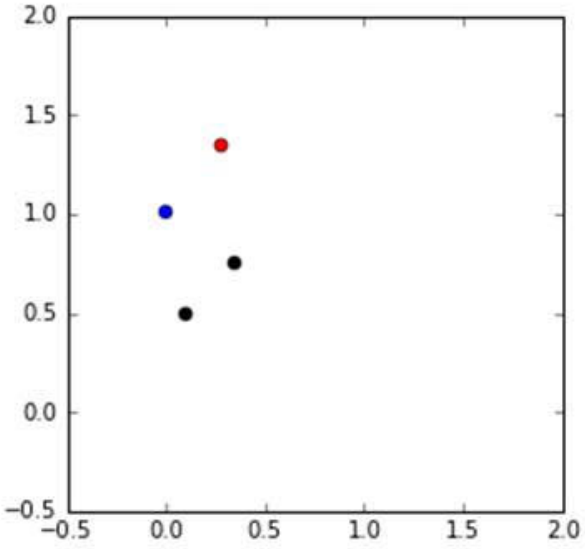


Figure 4: Unit spheres for norms l_1, l_2, l_∞



z2=[0.35,0.75] #black
z3=[0.28,1.35] #red
z4=[0,1.01] #blue

In bold in the lowest distance for the couple norm/distance, the clusters with the lowest distance will merge first.

Distance	Clusters	distance	l_1	l_2	l_∞
SL	Blue, Black	d(z1,z4)	**0.61**	0.5197	0.51
SL	Blue, Red	d(z3,z4)	0.62	**0.4405**	**0.34**
SL	Black, Red	d(z2,z3)	0.67	0.6041	0.6
CL	Blue, Black	d(z2,z4)	**0.61**	**0.436**	0.35
CL	Blue, Red	d(z3,z4)	0.62	0.4405	**0.34**
CL	Black, Red	d(z1,z3)	1.03	0.8688	0.85
AL	Blue, Black	[d(z1,z4)+d(z2,z4)]/2	**0.61**	0.4778	0.43
AL	Blue, Red	d(z3,z4)	0.62	**0.4405**	**0.34**
AL	Black, Red	[d(z1,z3)+d(z2,z3)]/2	0.85	0.7365	0.725

K-Means [15 pts]

Implement K-Means clustering from scratch.[1]. You have been provided with the MNIST dataset. You can learn more about it at http://yann.lecun.com/exdb/mnist/. The MNIST task is widely used in supervised learning, and modern algorithms with neural networks do very well on this task. We can also use MNIST for interesting unsupervised tasks. You are given representations of 6000 MNIST images, each of which are 28×28 handwritten digits. In this problem, you will implement K-means clustering on MNIST, to show how this relatively simple algorithm can cluster similar-looking images together quite well.

Problem 3 (K-means, 15pts)

The given code loads the images into your environment as a 6000x28x28 array. Implement K-means clustering on it for a few different values of K, and show results from the fit. Show the mean images for each class, and by selecting a few representative images for each class. You should explain how you selected these representative images. To render an image, use the numpy imshow function, which the distribution code gives an example of. Use squared norm as your distance metric. You should feel free to explore other metrics along with squared norm if you are interested in seeing the effects of using those. Also, your code should use the entire provided 6000-image dataset (which, by the way, is only 10% of the full MNIST set).

Are the results wildly different for different restarts and/or different K? Plot the K-means objective function as a function of iteration and verify that it never increases.

Finally, implement K-means++ and see if gives you more satisfying initializations (and final results) for K-means. Explain your findings.

As in past problem sets, please include your plots in this document. There may be tons of plots for this problem, so feel free to take up multiple pages, as long as it is organized.

Solution

We implemented the K-means algorithm on the data set. Since the algorithm converges to a solution and stops improving I tweaked the algorithm so that it stops as soon as the loss function converged to a minimum. We gain computational time this way. On the chart below we see the loss decreases strictly in function of the iterations. It converges in average in around 50 steps for K=10.

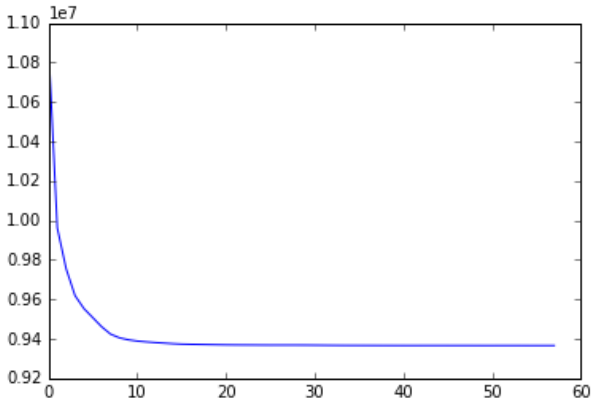


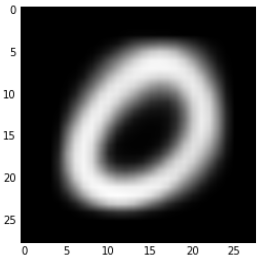
Figure 5: An example of convergence in 58 iterations

[1] That is, don't use a third-party machine learning implementation like `scikit-learn`; `numpy` is fine.

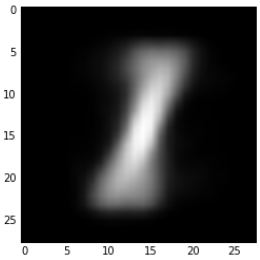
Since we are working with digits it made sense to start with K=10. Results were 'encouraging'. Unfortunately, the algorithm was not able to truly differentiate a 4 and a 5. Here are the mean images for the 10 clusters. Those mean images do not belong to the sample. Each image is define by 28*28 pixels and each pixel is a number for its darkness. The mean image is computed by taking the average of darkness for the 28*28 pixels. Since they are not actual handwritten image but averages, they are blurry. In the next pages, you will also find mean images for K=15 and 20. Finally, For K=10, we got 10 representative images for the first two clusters. We computed the distance to the nearest cluster and selected the 10 closest.

Figure 6: Mean images for K=10

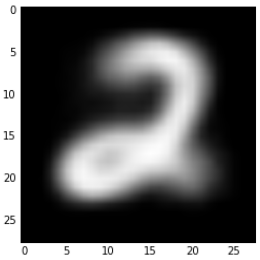
(a)



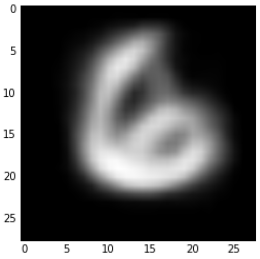
(b)



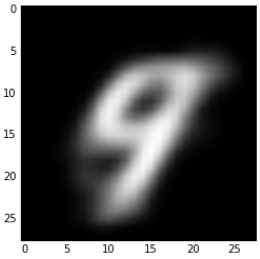
(c)



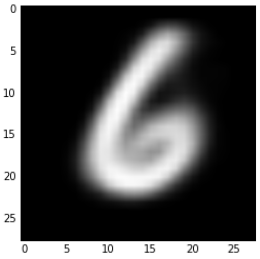
(e)



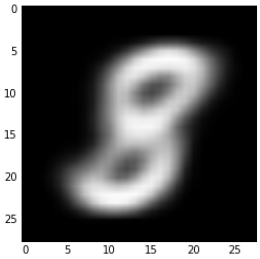
(f)



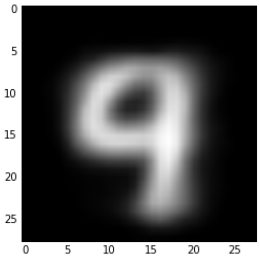
(g)



(i)



(j)



We finish with the K++ initialization. Below you will find the 10 images that will be used for the initialization of the K-mean algo. We ran it for K=10. We notice several '3' in this example but after a few iterations the algorithm lead to a clustering similar to what a simple K-mean would have given. Since the initialization provides already very "different" images in input, the clustering is faster and lead to results at least as good as the simple version. We notice a convergence in average in around 35 steps.

Figure 7: Mean images for K=15

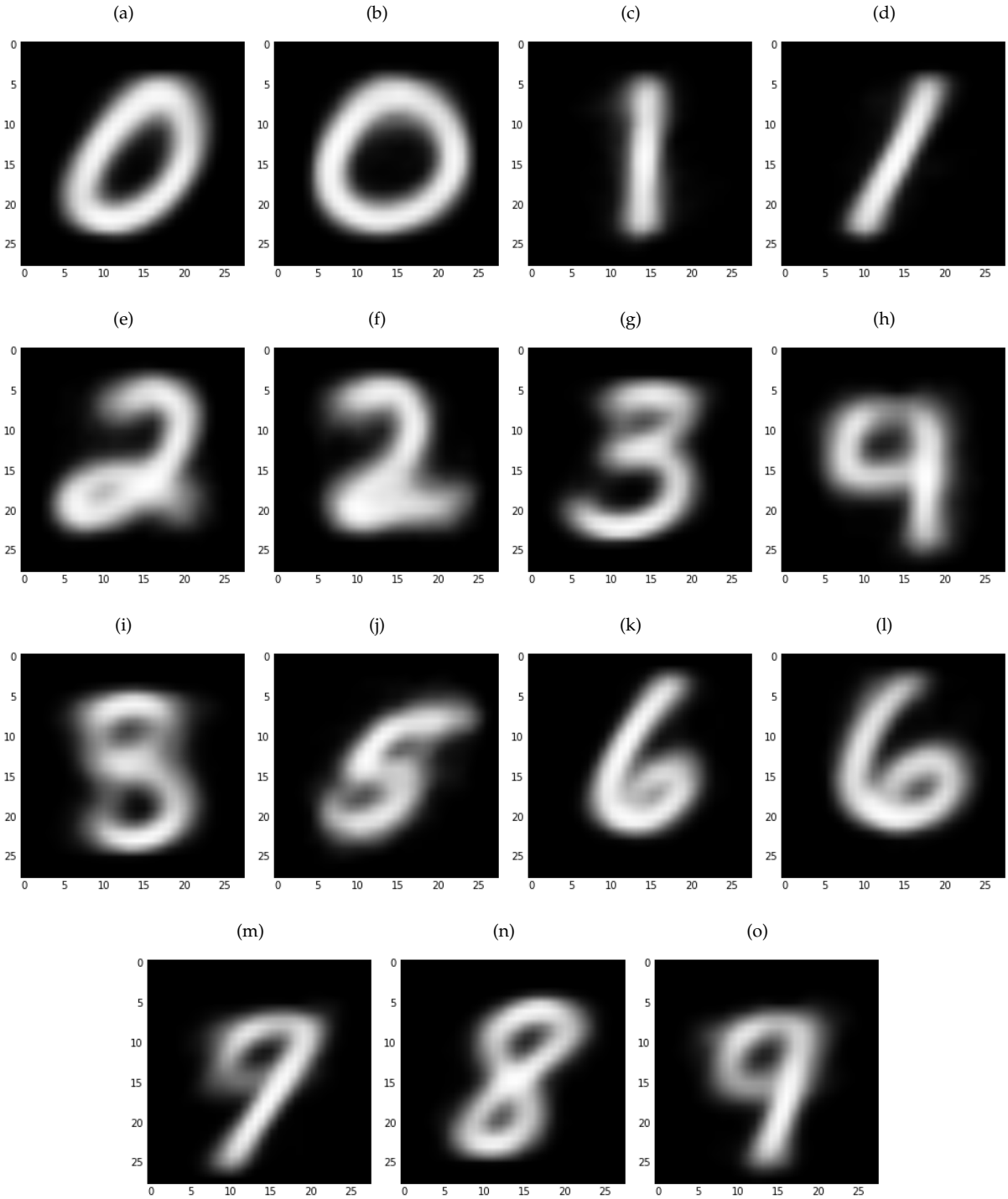
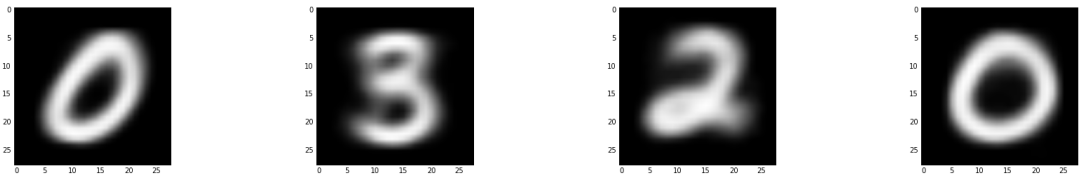
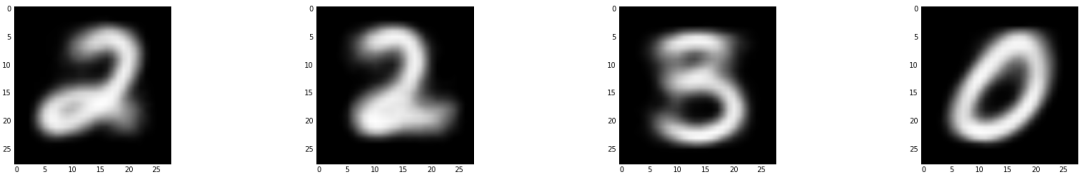


Figure 8: Mean images for K=20

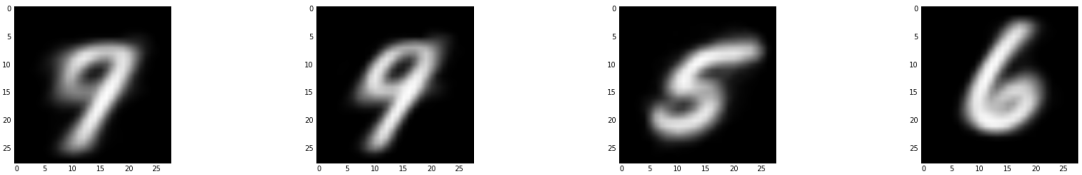
(a) (b) (c) (d)



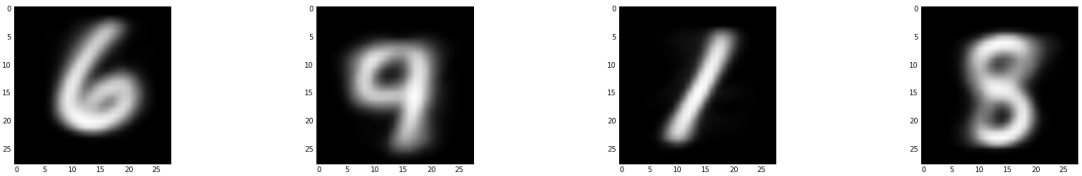
(e) (f) (g) (h)



(i) (j) (k) (l)



(m) (n) (o) (p)



(q) (r) (s) (t)

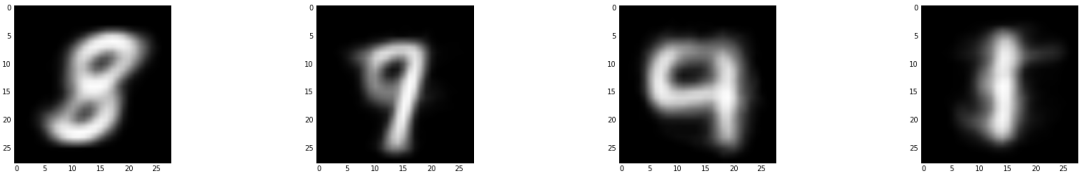


Figure 9: 10 most representative images for the cluster ’0’

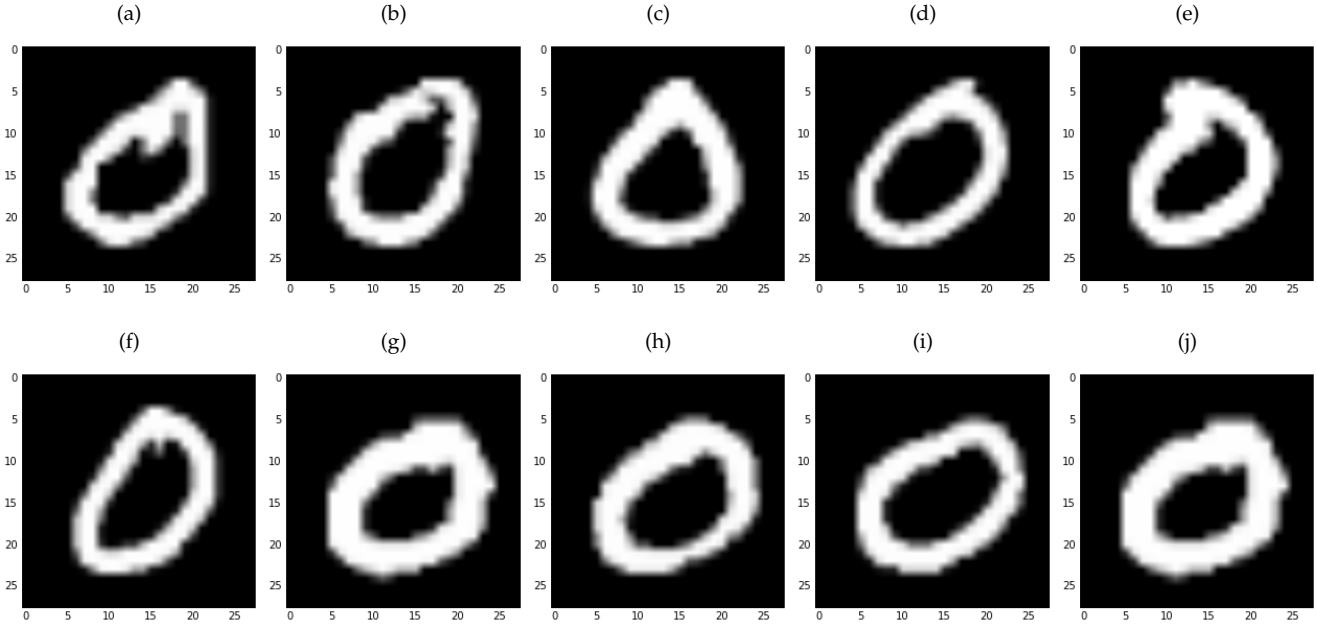


Figure 10: 10 most representative images for the cluster ’1’

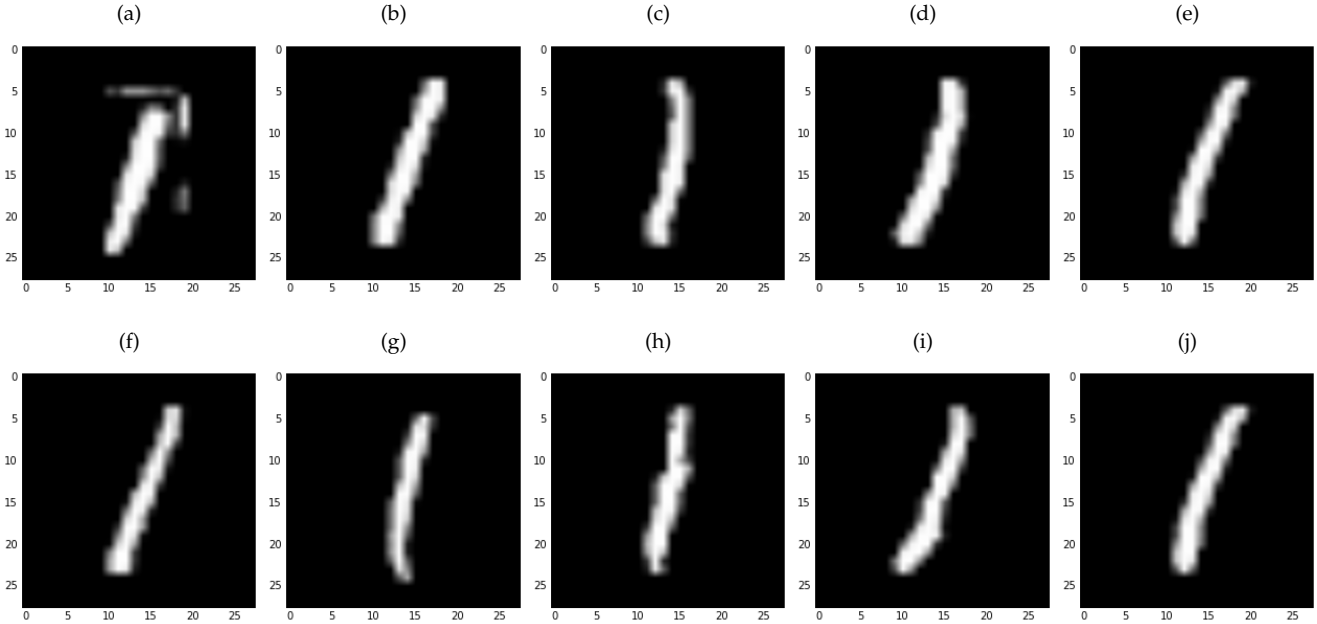
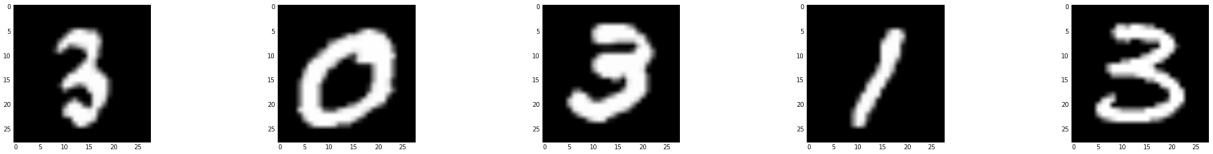
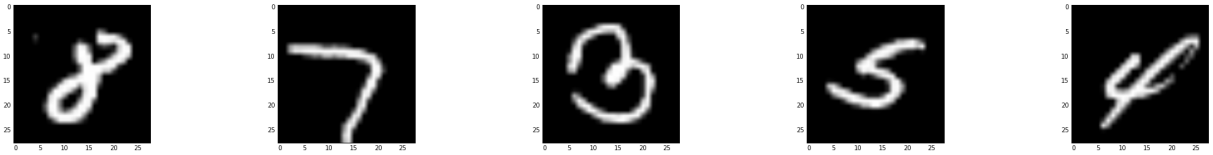


Figure 11: 10 images for the initialization of the K-mean algo (K-Mean++)

(a) (b) (c) (d) (e)



(f) (g) (h) (i) (j)



Problem 4 (Calibration, 1pt)
Approximately how long did this homework take you to complete? 12h