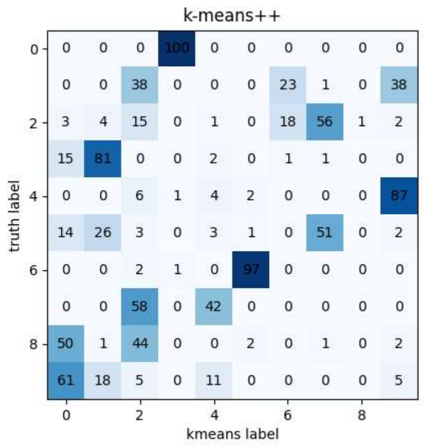
CS510: Deep Learning
Winter 2021, Assignment #4
Steve Braich

Source Code:
GitHub: https://github.com/steve3p0/cs510dl/tree/master/hw4

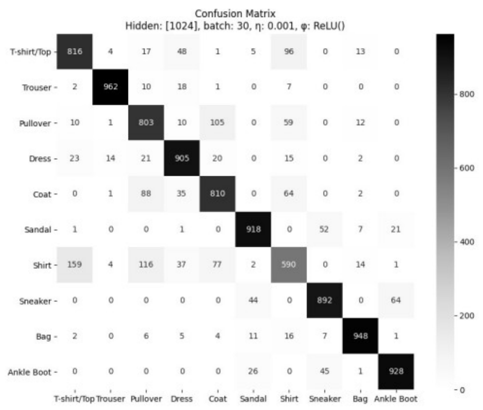
1. ***k-means clustering:***
From the MNIST data set, pick 100 samples from each of the 10 classes. Take all these 1,000 images and run them through a k-means clustering algorithm (k=10). You can use the scikit-learn library. Here is a link:
https://scikit-learn.org/stable/modules/clustering.html#k-means

The examples provided should be sufficient to enable you to write the code in python and run it.

    a. *What is the accuracy of the clustering? To measure accuracy, simply count the number of images from class a that are clustered into class b (for all a and b in 0…9). Produce a table of size 10x10. Each column and row is labeled 0-9. An entry (i, j) is a count of how many members from class i are clustered into cluster for class j. How would you compare the accuracy of this method with what you achieved previously in the first assignment?*



	0		2		4		6		8	
0	0	0	0	100	0	0	0	0	0	0
	0	0	38	0	0	0	23	1	0	38
2	3	4	15	0	1	0	18	56	1	2
	15	81	0	0	2	0	1	1	0	0
4	0	0	6	1	4	2	0	0	0	87
	14	26	3	0	3	1	0	51	0	2
6	0	0	2	1	0	97	0	0	0	0
	0	0	58	0	42	0	0	0	0	0
8	50	1	44	0	0	2	0	1	0	2
	61	18	5	0	11	0	0	0	0	5



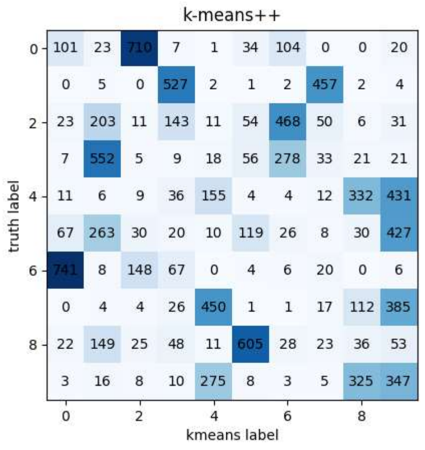
	T-shirt/Top	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle Boot
T-shirt/Top	816	4	17	48	1	5	96	0	13	0
Trouser	2	962	10	18	1	0	7	0	0	0
Pullover	10	1	803	10	105	0	59	0	12	0
Dress	23	14	21	905	20	0	15	0	2	0
Coat	0	1	88	35	810	0	64	0	2	0
Sandal	1	0	0	1	0	918	0	52	7	21
Shirt	159	4	116	37	77	2	590	0	14	1
Sneaker	0	0	0	0	0	44	0	892	0	64
Bag	2	0	6	5	4	11	16	7	948	1
Ankle Boot	0	0	0	0	0	26	0	45	1	928

The accuracy of the clustering is quite poor. Given that we are tasked with training with only 1000 samples, the poor performance is not surprising. We are not exactly comparing apples to apples here. On the first assignment, we built a classifier trained on the Fashion MNIST dataset to classify pictures of clothing. Here we built a model to cluster pictures of MNIST handwritten digits. Nevertheless, we achieved 90% accuracy on assignment #1. On this assignment, we can see that due to the low number of training samples, our results are quite poor.

2. ***k-means clustering using feature vectors:***
This time we will take 1,000 images from each of the 10 MNIST classes giving us 10,000 images. Build an autoencoder and train it. You can use the code in the second part of this page: https://scikit-learn.org/stable/modules/clustering.html#k-means

After training, you obtain the feature vectors by using the encoder part alone (model.encoder(x)). Then use k-means clustering of these feature vectors.

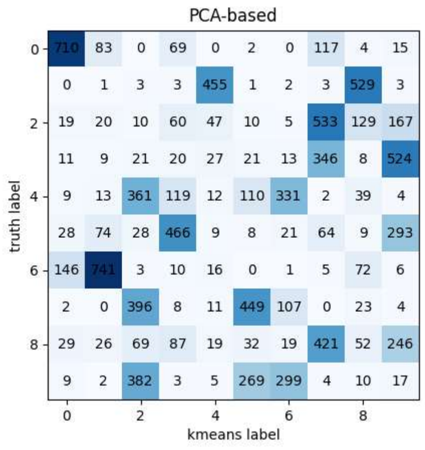
a. *As above, produce a table and then explain any differences between these results and the ones obtained in part 1 above.*


k-means++
truth label
kmeans label
0
2
4
6
8

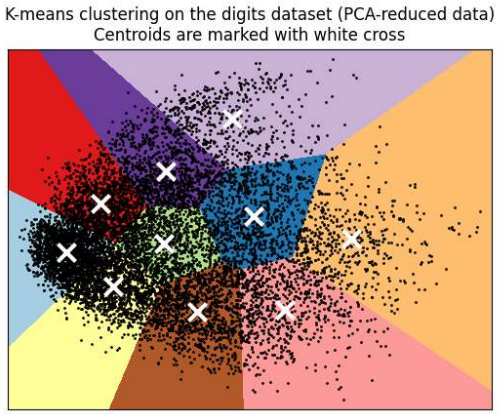
101	23	710	7	1	34	104	0	0	20
0	5	0	527	2	1	2	457	2	4
23	203	11	143	11	54	468	50	6	31
7	552	5	9	18	56	278	33	21	21
11	6	9	36	155	4	4	12	332	431
67	263	30	20	10	119	26	8	30	427
741	8	148	67	0	4	6	20	0	6
0	4	4	26	450	1	1	17	112	385
22	149	25	48	11	605	28	23	36	53
3	16	8	10	275	8	3	5	325	347

It looks like we have some improvement autoencoding the images before we cluster them using k-means, but not by much. I'm wondering how much of this improvement is due to a larger training dataset (10,000 vs. 1000) than to using an autoencoder.

b. *(Bonus 5 points) Run PCA on the feature vectors, reduce the dimensionality to just a few most important dimensions and then do the clustering.*


PCA-based
truth label
kmeans label
0
2
4
6
8

710	83	0	69	0	2	0	117	4	15
0	1	3	3	455	1	2	3	529	3
19	20	10	60	47	10	5	533	129	167
11	9	21	20	27	21	13	346	8	524
9	13	361	119	12	110	331	2	39	4
28	74	28	466	9	8	21	64	9	293
146	741	3	10	16	0	1	5	72	6
2	0	396	8	11	449	107	0	23	4
29	26	69	87	19	32	19	421	52	246
9	2	382	3	5	269	299	4	10	17



The only noticeable difference, was the time it took to train. On my GPU, it took about five seconds to cluster with k-means vs. about one second with PCA. This doesn't include training the Autoencoder and running images thru to get features.