COMP3105 Assignment 3

Mario Pardo 101286566 Dante Farinon-Spezzano 101231566

Q1d

Training Accuracies

n	Model 1	Model 2
16	0.9875	0.98125
32	0.878125	0.953125
64	0.875	0.9546875
128	0.84453125	0.94140625

Testing Accuracies

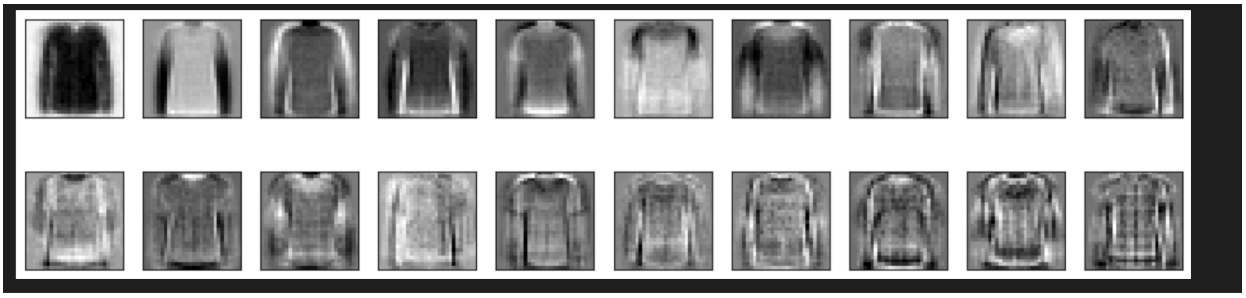
n	Model 1	Model 2
16	0.7055	0.8843
32	0.7879	0.8851
64	0.8189	0.901
128	0.8365	0.9183

At a quick glance, we see that our training accuracies are higher than our testing. This is to be expected.
For both training and testing, model 2 performs better. This could be due to the way that the data is generated, making it fit better with our classification techniques.
In our training accuracies, we see that as n grows, our training accuracies seem to shrink while the testing accuracies grow.

For training, it will be easy to train our model to easily fit small amounts of data, but then as we get more data, it is harder and harder to fit. This is why we see training accuracies shrink as n grows.

In our testing accuracies, we see that a larger n values makes our accuracies increase.
This is due to the fact that we trained our model on a larger amount of data, so it will be better at extrapolating to other unseen data.
As n grows, the pool of data on which a model is trained on becomes more representative of all sorts of (random) data.

Q2.b)



We can see that the 5th shirt matches the one from the assignment specifications, and the first few eigen-shirts capture general shirt shapes and outlines. Some eigen-shirts show textures or patterns, such as vertical or horizontal stripes. The outlines of the shirts are shown more in the pictures above, like in the third photo, where the white outline goes around the shirt.
Since these shirts are basically our “basis” shirts for all shirts, this makes sense since these are basic features that we can expects some shirts to have or not. They are like our “building blocks” for all shirts, so we should expect to have shirts with features that other shirts might or might not have.
Most eigen-shirts appear with inverted colors, as we were told to potentially expect. The resulting shirts seem to be more pixelated, grainy versions of the originals - this makes sense since we are doing dimensionality reduction.

2d)

Average Training Accuracies

Dimension (k)	Model	Avg Training Accuracy
1 (i=0)	1 (j=0)	0.8489
1 (i=0)	2 (j=1)	0.6459
2 (i=1)	1 (j=0)	0.8593
2 (i=1)	2 (j=1)	0.9409

Average Test Accuracies

Dimension (k)	Model	Avg Test Accuracy
1 (i=0)	1 (j=0)	0.8431
1 (i=0)	2 (j=1)	0.6313
2 (i=1)	1 (j=0)	0.8378
2 (i=1)	2 (j=1)	0.9165

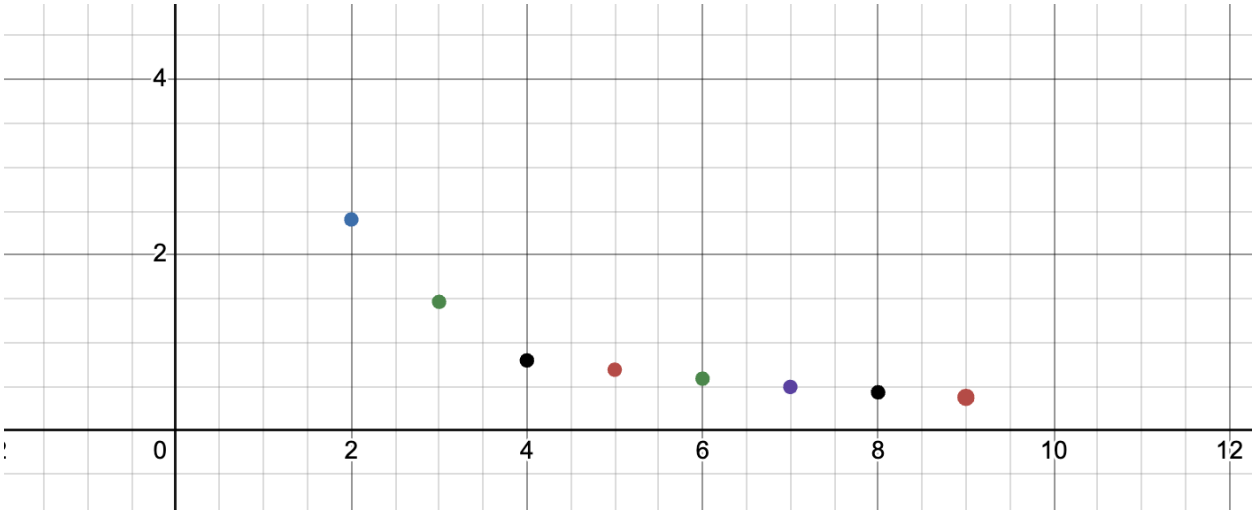
2e)
General assessment:
For Model 2, increasing the PCA dimension from 1 to 2 leads to a significant improvement in both training accuracy (from 63% to 91%) and test accuracy (from 64% to 94%). This indicates that the data generated by Model 2 needs at least 2 dimensions to capture the key information needed to classify, due to it having a more complex structure. All the other values are straightforwardly accurate, and don't change from training to test, which shows that it doesn't overfit to the training data.

Model Differences: gen_model=1 performs reasonably well with both 1 and 2 dimensions, while gen_model=2 needs 2 dimensions to achieve high accuracy.

Generalization: The model shows similar training and test accuracies across configurations, indicating good generalization without significant overfitting.

3c)

K value	2	3	4	5	6	7	8	9
obj-value	2.404	1.465	0.7974	0.6915	0.5893	0.4951	0.4346	0.3765



Using the elbow method, we see that the squared differences start to decrease less past k = 4, so the trade off between using a higher k-value and trying to minimize the difference also tails off. This would lead me to pick k = 4 or k = 5 as it gives us the best of both worlds (still a small difference but not a super high k-value that isn't impacting the minimization of the difference, and avoids overcomplicating with additional clustering of the data)

```
#3c

def chooseK(X, k_candidates=[2, 3, 4, 5, 6, 7, 8, 9]):

    obj_val_list = []

    for k in k_candidates:
        # Call repeatKmeans with X and k, and retrieve only the best objective value
        _, _, obj_val = repeatKmeans(X, k)
        obj_val_list.append(obj_val)

    return obj_val_list

Xtrain, Ytrain = generateData(n=100, gen_model=2)
obj_val_list = chooseK(Xtrain)

k_candidates=[2, 3, 4, 5, 6, 7, 8, 9]

print("Objective values for different k:")
print("k:", k_candidates)
```

```
print("Objective values:", obj_val_list)
```