COMP3105: Assignment 3

Alexander Breeze 101 143 291
Victor Litzanov 101 143 028

Environment:

- Windows 10 20H2 x64
- Python 3.10.7
- Numpy 1.23.3
- Scipy 1.9.1
- Matplotlib 3.6.0

Solutions:

1e)

Train

n	Model 1	Model 2
16	94%	99%
32	87%	97%
64	86%	97%
128	86%	95%

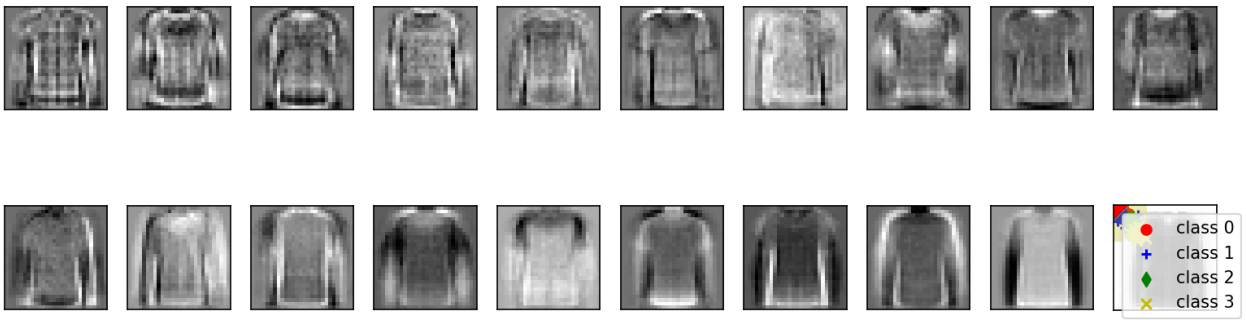
Test

n	Model 1	Model 2
16	78%	86%
32	81%	89%
64	81%	90%
128	84%	92%

The accuracies are higher in model 2 than in model 1, however this is due mostly to model 2 plotting points over the entire space instead of focusing on a diagonal strip as in model 1. This increase in group size means most points are well within their groups, and less are outliers invading other groups. This is also reflected in the training data, as for lower n the accuracies are higher. This is due to the model overfitting on the training data, which is easier to do when n is smaller as that means less data points and so less outliers to have to accommodate. In the test data the inverse is true, which is odd until you account for the fact that the tests with larger

n are on models trained with larger n. This explains the result, as larger training datasets means a more even distribution of outliers and thus the overfitting on any one outlier being canceled out by overfitting on other outliers, resulting in an even fit overall.

2b)



The eigenshirts are “blurrier” than the original images, e.g. instead of hard borders between colors it slowly fades from one to another via multiple pixels with multiple shades of gray. This is similar to squinting, where your eye absorbs significantly less incoming photons and thus things become similarly blurry. By replacing the images with eigenvectors we represent them as simpler waves instead of hard values for each individual pixel, allowing for greater compression of data.

2d)

Train

Dim k	Model 1	Model 2
1	85%	65%
2	85%	95%

Test

Dim k	Model 1	Model 2
1	85%	63%
2	83%	91%

2e)

As model 1 constrains the data to one strip, it can be effectively modeled with k=1 dimensions Model 2 puts the data into 4 quadrants, 2x2, so it requires k=2 dimensions to properly model. That is why model 1 has the same accuracy for k=1 and k=2, whereas model 2 receives a significant increase in accuracy when going from k=1 to k=2 dimensions. Accuracies decrease slightly when going from training to test data, but the decrease is small, indicating the models were not significantly overfit.

3c)

k	2	3	4	5	6	7	8	9
obj_val	1.222	1.231	0.696	0.696	0.688	0.696	0.696	0.696

I believe the optimal value of k is 6, since it did slightly better than the next best values. I think that the result of 6 doing better than others is a fluke as in most other runs all results for k>3 were identical, due to them all finding the same global minimum. However, in some cases 6 did outperform others, so it does at least as good and sometimes better, making it the optimal choice.