Assignment 4

Johnnie Oldfield

October 2019

1 Problem Definition

The purpose of this assignment is to use a previously implemented memory-based deterministic classifier using nearest neighbor but in a subspace computed though Principal Component Analysis (PCA). The classifier can only handle 1-NN(nearest neighbor). The goal is to observe how reducing the dimensional of the original data points to see how it affects the accuracy of the classifier. I will be using the Modified National Institute of Standards and Technology(MNIST) data set of handwritten numbers.

2 Decisions

In order the reduce the dimensionality of the data points I must first find the means of both the testing and training sets and subtract the sets by their means. Using trn_size and tst_size I can fix the number of samples for training and testing. The main experiment will be using 10,000 training and 1,000 testing samples. See listing 1.

Listing 1: Setting Up the Dataset into means with a fixed size.

```
trn_size = 10000
tst_size = 1000
trainX = trainX[:trn_size, :]
testX = testX[:tst_size, :]
xzero = trainX - np.mean(trainX, axis=0)
test_zero = testX - np.mean(testX, axis=0)
y = trainy[:, :trn_size][0]
```

I decided not to shuffle the data set. The goal is to test how the dimensionalilty affects the accuracy based on the number of components.

Listing 2: Getting the components of training set.

```
components = {}
[u, s, v] = np.linalg.svd(xzero)
PCs = v.T
```

In listing 2 I initialize a components dictionary to keep track of each component and the accuracy it gets when the NN classifer is ran in that space. To get the principle components I used numpy's liner algebra function svd or Singular value decomposition [1].

Listing 3: Projecting Testing/Testing and storing accuracy's

```
for comps in range(1, 31):
    Xprojected = np.matmul(xzero, PCs[:, :comps])
    Tprojected = np.matmul(test_zero, PCs[:, :comps])
    components.update( {comps : NN(Xprojected, Tprojected)} )
```

I used a for loop to iterate through each component and created projected spaces to be used by the NN function. The components dictionary is updated with the number of components as the key and the accuracy of the classifier in that subspace, which is returned by the NN function.

Listing 4: NN function

```
def NN (Xprojected, Tprojected):
  ec = 0
  for tstx in range(tst_size):
    edistances = []
    for trnx in range(trn_size):
      edistances.append( euclidean_distance(
        Xprojected[trnx, :], Tprojected[tstx, :] ) )

    edistances = np.array(edistances)

    if(y[np.argmin(edistances)] == testy[0][tstx]):
      ec += 1
  return (ec/tst_size)
```

The NN function accepts the projected spaces as parameters. It starts with ec to keep count of correct classifications then iterates though the test

range then the training range to call the distance function and append it to the array. If the max of that array is equal to the class in the test y at the current test index then it is classified correctly and the count is incremented. After the nested loop is finished the accuracy is returned.

Listing 5: Distance function

```
1 def euclidean_distance(x, y):
2   return np.sqrt(np.sum(np.power(x-y, 2)))
```

Listing 5 shows the functions that was used to get the Euclidean distance. Euclidean distance for n dimensions can be found by using the following equation [2]:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

3 Evaluation

For the assignment I ran multiple different experiments changing the size of the training samples and testing samples. Increasing them from the main experiment of 10,000 and 1,000 did not do anything interesting so I excluded them.

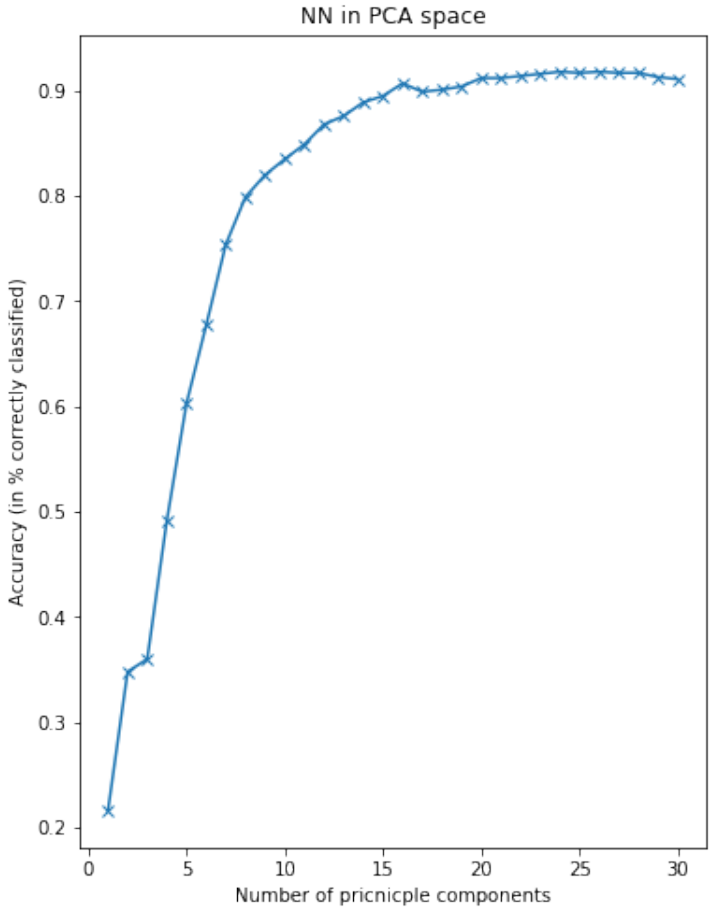

Figure 1: Using 5,000 testing

In the main experiment the accuracy steadily rises until it plateaus around 20 components with an accuracy 90%. Even with just 10 components an accuracy of around 82% is not bad at all.

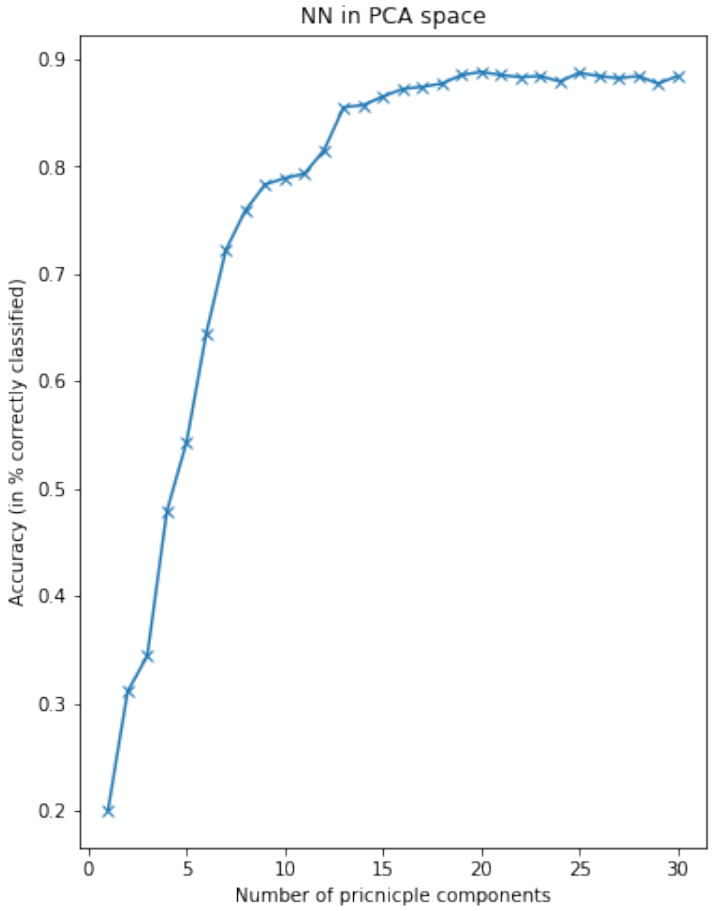

Figure 2: PCA using 2,000 training samples and 1,000 testing samples.

Very similar to the test in Figure 1 this experiment plateaus around the 15-20 component range. The only real difference being the dip at 8-12 where the accuracy fell off before catching up to the peak and the overall accuracy peaked below 90%.

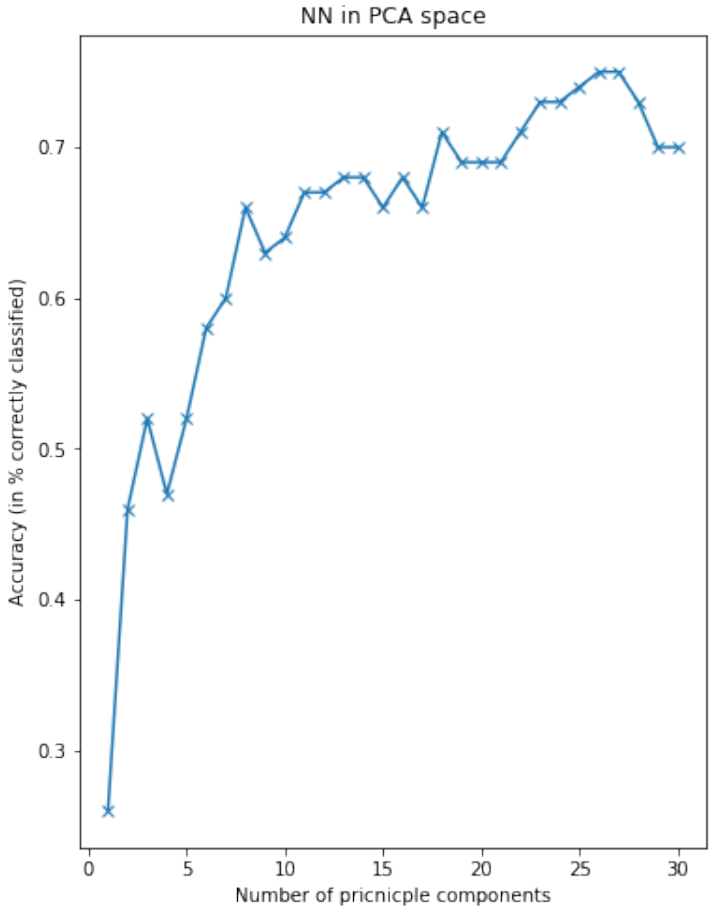

Figure 3: PCA with only 200 training and 100 testing.

Setting the sample sizes to such an extreme is one of the few ways to get any significant change. Although, the accuracy decrease is namely due to the lack of training the classifier received than the components, the accuracy does not plateau as the other experiments did.

4 What I Learned

I learned a lot on how principle component analysis works. I assumed you need most of the data (if not all) to derive anything useful from it. But this assignment has shown otherwise. With only 20 or so components the classifier was able to identify digits very accurately.

5 Future Work

The classifier can currently only get 1-NN. This can be expanded into a K-NN classifier. IT would be interesting to see how different K's can affect the accuracy in PCA space. The PCA is also being tested with NN using only euclidean distance. Further experimentation can compare how different distances are influenced in PCA space (e.g Euclidean vs Cosine).

References

[1] `https://en.wikipedia.org/wiki/Singular_value_decomposition`.

[2] `https://en.wikipedia.org/wiki/Euclidean_distance`.