Implementation of Principal Component Analysis (PCA) and Understanding PCA with MNIST Dataset

Sami Alperen Akgun
sami.alperen.akgun@gmail.com

A. Implementation of PCA

To calculate PCA for a given matrix in size DxN, a python function named f_{PCA} was created:

```
def f_PCA(input_data,d):
        # input_data = matrix DxN where D=# dimensions and N=# of samples
        # d = It decides output dimension --> output=dxN
        # output: input_data but dimension is d --> so output size dxN
        # output 2: W matrix --> which is useful for inverse PCA
        # Step 1
        mu= np.mean(input_data,axis=1,keepdims=True) #mu= Dx1
        cov= np.cov(input_data) #cov= DxD

        # Step 2
        input_data_hat = input_data- mu #input_data_hat has zero mean --> shape DxN

        # Step 3
        Evalues, Evectors= np.linalg.eig(cov)

        # Step 4
        # Select the largest d eigenvalues
        # It is sorted, so principal_eigenvalues[0] >  principal_eigenvalues[1] > ... >
        p_evalues_col_indices = Evalues.argsort()[::-1][:d]
        W = Evectors[:,p_evalues_col_indices]

        # Step 5
        y = np.dot(W.T,input_data_hat) #matrix multiplication
        return y, W
```

B. Selecting a Suitable Dimension d Corresponding to Proportion of Variance (POV) 95%

Proportion of Variance (POV) can be calculated for a given d value using Equation 1.

$$POV = \frac{\sum_{i=1}^{d} \lambda_i}{\sum_{i=1}^{D} \lambda_i} \text{ , where D is the total number of features in original data} \quad (1)$$

To find the suitable d for POV value 95%, POV value was calculated iteratively for different d values starting from 1. Due to numerical errors, calculated POV value can not be exactly equal to 95%. Therefore, a small ϵ value was used to determine whether calculated POV is suitable for 95% POV value. One can see the implemented algorithm below:

```
cov = np.cov(X) #cov= DxD
eigenvalues, _ = np.linalg.eig(cov)
total_of_all_eigenvalues = np.sum(eigenvalues)
desired_pov = 95/100 #percent
eigenvalues_sorted = -np.sort(-eigenvalues) #ascending order, i.e. first elements is the largest

calculated_pov = 0
epsilon = 0.001
# calculated_pov - desired_pov might be not exactly same, it is good practice to
# put some epsilon value
eigenvalue_sum = 0
d = 0 #counter for while loop, also our d value for PCA algorithm
while abs(calculated_pov - desired_pov) > epsilon:
        eigenvalue_sum += eigenvalues_sorted[d]
        calculated_pov = eigenvalue_sum / total_of_all_eigenvalues
        print("Desired pov is ", desired_pov)
        print("Calculated pov is ", calculated_pov)
        d += 1
```

```
# Since python indices start from zero, we need to add 1 after the loop
# to get the real d value
d += 1
print("d value corresponding to ", desired_pov, " is ", d)
```

When ϵ was selected as 0.001:

```
Calculated pov is  0.9492863456560553
d value corresponding to 0.95% POV is 153
```

C. Selecting a Suitable Dimension d Considering Average Mean Square Error (MSE)

For different values of d=1, 20, 40, 60, 80, ..., 760, 784, calculated average mean square error (MSE) can be seen in Figure 1.

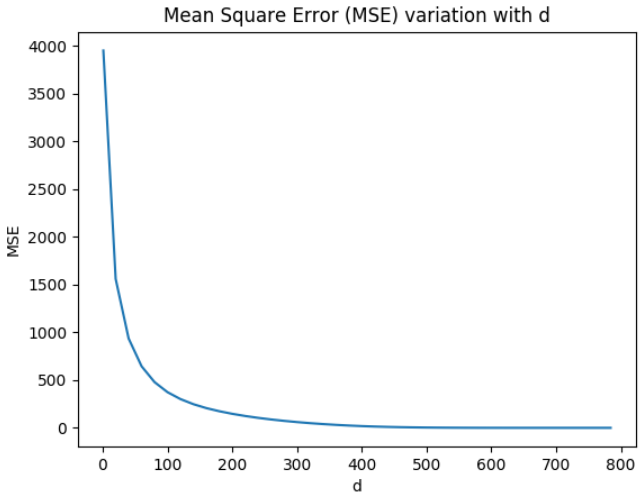


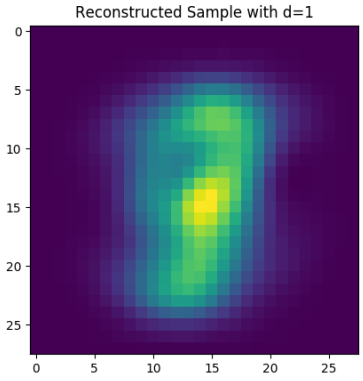
Fig. 1. The Average Mean Square Error (MSE) vs different values of d

From the plot, it can be said that around d value 100, calculated MSE starts to converge abruptly to zero. After d value 300, there is almost no significant change in calculated MSE with increasing d value. Therefore, we can clearly say that selecting d bigger than some certain value will not affect the performance significantly. It will just require additional computational power. In addition, we can claim that our proposed d value 153 for POV 95% is actually provide quite low MSE value. Therefore, selecting d value as 153 might be a wise choice to balance the trade-off between MSE and computational power with increasing d value. One can see PCA reconstruction algorithm $f_{inv-PCA}$ and it is usage below:

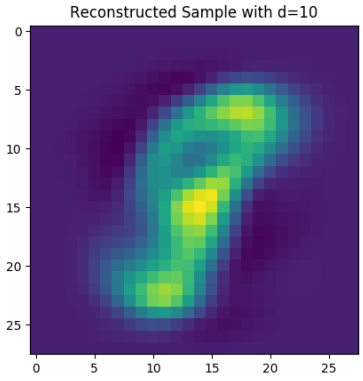
```
def f_inv_PCA(input_data,W):
        # input_data = matrix dxN where d=# reduced dimensions and N=# of samples
        # W = size dxD matrix containing all eigenvectors of original data
        # output: original data size dxN, but with zero mean

        reconstructed_data_hat = np.dot(W,input_data)
        return reconstructed_data_hat

reduced_d, W = f_PCA(X,d_value)
reconstructed_d = f_inv_PCA(reduced_d,W) + mu
```

D. Reconstruction of Original Image

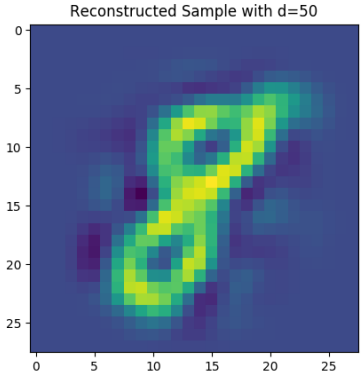
For different values of d=1, 10, 50, 250, 784, reconstructed sample from the class of number 8 can be seen in Figure 2. From this figure, we can again deduce the fact that selecting d larger than a specific value or close to the original value don't make any significant difference in reconstructed samples like in the case between d=250 and d=784. On the other hand, we can also state that selecting d smaller than the suitable value might harm the learning algorithm. For instance, for d=1 and d=10, the reconstructed samples are so blurry that we as humans even can not understand which class they belong to. Lastly, for d=50 case, the reconstructed sample might be okay for a sophisticated classification algorithm, but may not be good enough for a simple one.



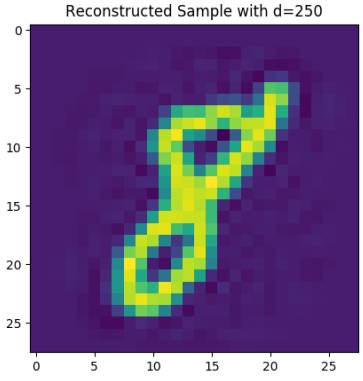
(a) d=1



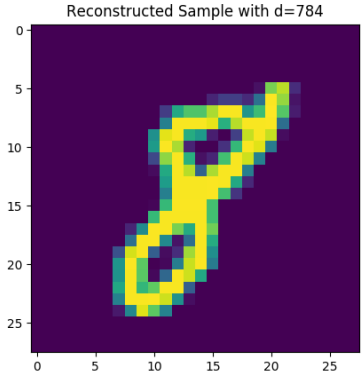
(b) d=10



(c) d=50



(d) d=250



(e) d=784

Fig. 2. Reconstructed sample from the class of number 8 for different d values

E. Analyze Variation of Eigenvalues with Reduced Dimension d

Variation of eigenvalues with d values can be seen in Figure 3. This plot basically shows the sorted version of eigenvalues in an ascending order. From the plot, we can conclude that after a specific d value (around 40), the difference between eigenvalues are subtle. Therefore, if we take the eigenvalues which are quite larger than others (first 40 principals), we will be able to reconstruct the images quite successfully without dealing with all the features. One can also realize that this plot looks similar to Figure 1 which shows MSE variation with varying d values. This should be indeed the case since when we capture largest eigenvalues (principals) which contain most of the necessary information, reconstruction error will be reduced significantly.

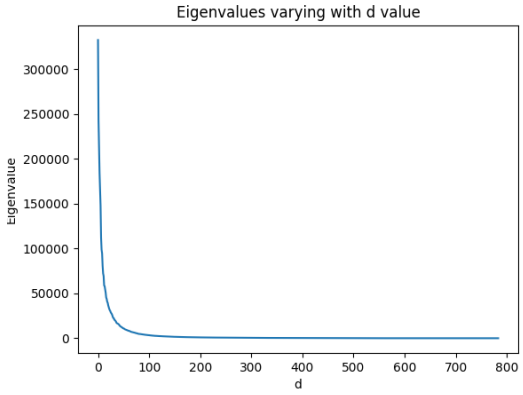


Fig. 3. Variation of largest eigenvalues with varying d