HW2

June 29, 2024

##

Johns Hopkins University

##

Whiting School of Engineering

##

Engineering for Professionals

##

685.621 Algorithms for Data Science

##

Homework 2

##

Assigned at the start of Module 3

##

Due at the end of Module 5

##

Total Points 100/100

Class, the below is a standard set of instructions for each HW, in this assignment groups will be set up for collaboration. Make sure your group starts one thread for the collaborative problems. You are required to participate in the collaborative problem and subproblem separately. Please do not directly post a complete solution, the goal is for the group to develop a solution after everyone has participated. Please ensure you have a write-up with solutions to each problem and subproblems, you are also required to submit code that will be compiled when grading the assignment.

1 1. Module 3 Note this is not a Collaborative Problem

30 Points Total In this problem use the seaborn Python package to create a scatter plot matrix of the Iris data set. 1. [10 points] Examine the Seaborn plots provided down the diagonal and present a clear and concise explanation for these plots.
2. [10 points] Does any feature separate the three classes? If so, which feature? If not, which is the

closest? 3. [10 points] Is there any combination of features that can separate the three classes? If so, which features? If not, which are the closest?

Reference: https://seaborn.pydata.org/examples/scatterplot_matrix.html

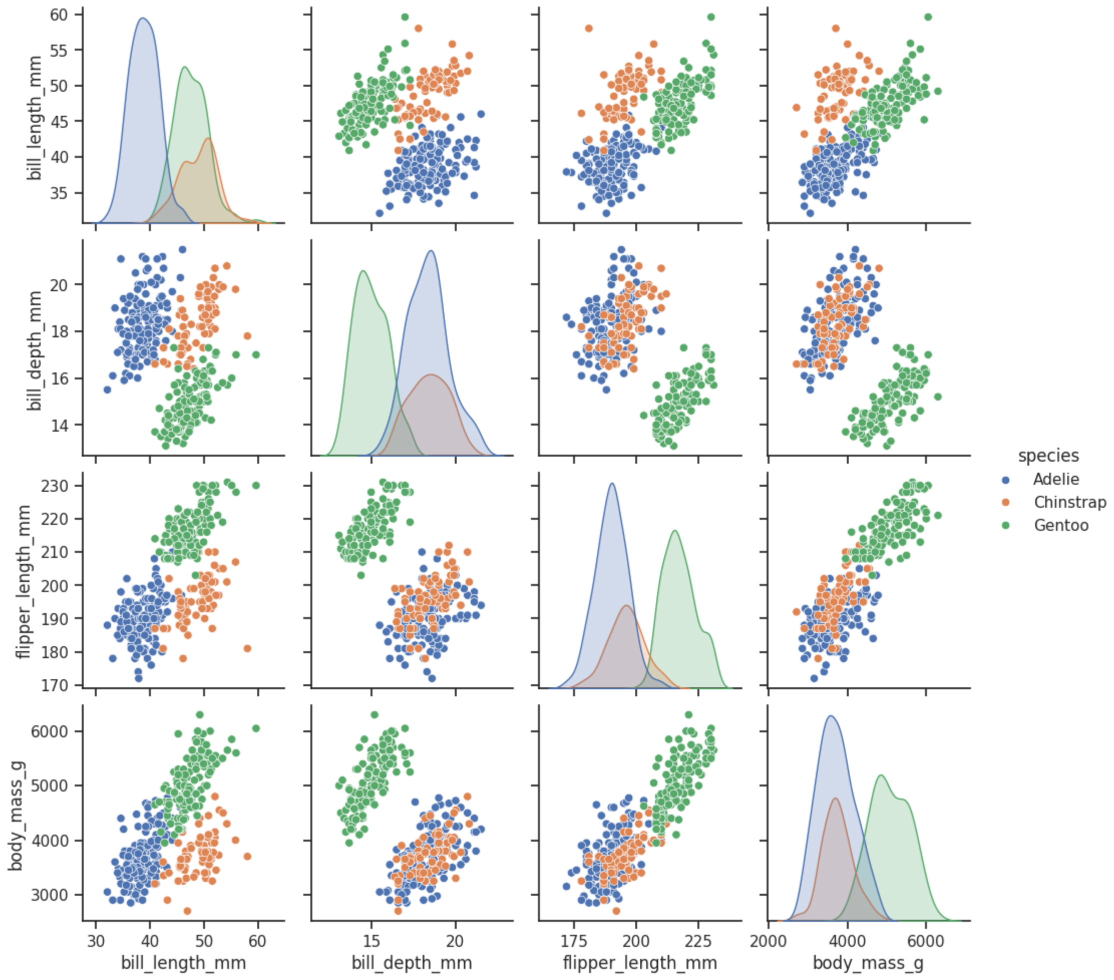
[ ]:
```python
## import libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from copy import deepcopy
import pywt

## import iris dataset
iris_df = pd.read_csv("iris.csv")
```

[ ]:
```python
## Type code here for the part 1 ##
import seaborn as sns
sns.set_theme(style="ticks")

df = sns.load_dataset("penguins")
sns.pairplot(df, hue="species")
```

[ ]:
```
<seaborn.axisgrid.PairGrid at 0x770542417cd0>
```



1.0.1 Examine the seaborn plot provided down the diagonal and present a clear and concise explanation for these plot.

Down the diagonal we see three colors of plots: blue, orange, and greem. Respectively these colors correspond to different species: adelie, chinstrap, and gentoo. The plots represent the density of each singluar feature in it's respective scales – compared to a pair of features in the off-diagonals. For instance we see that the first plot in the diagonal helps to show the density of all the datapoints for the `bill_length_mm`. This will tell us the concentration of the feature based on each species. For `bill_length_mm`, we know that the largest number of datapoints belong around 60mm for the Adelie species. Additionally, we know around 52mm and 42mm respectively for Chinstrap and Gentoo. Further down the diagonal this holds for `bill_depth_mm`, `flipper_length_mm`, and `body_mass`.

1.0.2 Does any feature separate the three classes? If so, which feature? If not, which is the closest?

No, we see that down the diagonal there is significant overlap for the `Chinstrap` class in each of the features. In each of these plots we do see a significant difference between the `Adelie` and `Gentoo` classes. Although not significant, `flipper_length_mm` is probably the most distinct feature because there is less overlap for the `Chinstrap` class

1.0.3 Is there any combination of features that can separate the three classes? If so, which features? If not, which are the closest?

Yes, the separate of the classes lie in the separation between the scatterplot. Each of the `bill_length_mm` plots, first row, seem to show adequate separate between each of the classes. `bill_depth_mm` vs. `bill_length_mm` also shows adequate separation as well. Lastly, `body_mass_g` vs. `bill_length_mm` also is pair of features that can separate the classes

2 2. Module 4 Note this is a Collaborative Problem

30 Points Total In this problem, implement code to use the Mahalanobis Distance on the Iris dataset. You may NOT use the built-in function in Python. 1. [10 points] Using all four features, calculate the 50 distances for the Setosa class. 2. [5 points] Calculate the range (min and max) of the 50 Mahalanobis distance values. What is the delta between the min and the max? 3. [10 points] If you remove the observation with the largest distance, do the remaining distances appear to be closer together? After removing the observation with the largest distance, recalculate the ranges with the updated mean and covariance. Does removing the observation with the largest Mahalanobis distance provide a smaller range between the min and max? 4. [5 points] Given your findings, describe what was accomplished following the method above this data point from the dataset.

Part 1 First we can filter the dataframe to only see the `setosa` species. Also we would only like to see the values and filter out the column `species` for data manipulation

[ ]:
```python
## Type code here for part 1 ##
setosa_df = iris_df[iris_df["species"] == "setosa"]
setosa_df = setosa_df.drop(columns=["species"])
setosa_values = setosa_df.values
```

We can now create our own homegrown implementation of the mahalanobis distance after calculating the mean, covariance, and inverse covariance of the passed data. We will use these values to follow the formulation below:

$$mahalanobis = \sqrt{(x-\mu)\Sigma^{-1}(x-\mu)^T}$$

[ ]:
```python
def create_mahalanobis_distance(df_values):
    data = deepcopy(df_values)
    n = data.shape[0]
    p = data.shape[1]
    mu = np.mean(data, axis=0, keepdims=True)
    #rowvar change since observation are in the rows
    cov = np.cov(data, rowvar=False)
    inv_cov = np.linalg.inv(cov)
    # compute the squared malanobis distance, cap the minimum to only positive␣
↪values
    mahalanobis_distance = np.sqrt(np.maximum((data - mu) @ inv_cov @ (data -␣
↪mu).T,0)).diagonal()
    return mahalanobis_distance
```

[ ]:
```
setosa_mahalanobis = create_mahalanobis_distance(setosa_values)
setosa_mahalanobis
```

[ ]:
```
array([0.66698442, 1.39300256, 1.14590085, 1.29205482, 0.9252263 ,
       1.91920834, 1.85193573, 0.54070921, 1.72628655, 1.66478036,
       1.35215307, 1.42359509, 1.61069941, 2.68373139, 3.20044667,
       2.73111366, 2.34944412, 0.80175606, 2.26334089, 1.28495791,
       2.22859256, 1.61362928, 3.34666234, 2.72761422, 3.09786973,
       1.85747736, 1.59756697, 0.86192245, 1.11047426, 1.45059611,
       1.34678509, 2.25139998, 2.8681089 , 2.3447252 , 1.66478036,
       1.80985963, 2.37067817, 1.66478036, 1.82105861, 0.70197087,
       1.29188685, 3.5536989 , 2.08163126, 3.46376956, 2.93230434,
       1.50040864, 1.70256269, 1.23882101, 1.11328423, 0.65873036])
```

Let's check this against scipy's implementation of the mahalanobis distance!

[ ]:
```
from scipy.spatial import distance
mean = np.mean(setosa_values, axis=0)
covariance_matrix = np.cov(setosa_values, rowvar=False)
inv_covariance_matrix = np.linalg.inv(covariance_matrix)

mahalanobis_distance = [distance.mahalanobis(observation, mean,␣
↪inv_covariance_matrix) for observation in setosa_values]
mahalanobis_distance
```

[ ]:
```
[0.6669844177020008,
 1.3930025560574884,
 1.1459008507832444,
 1.2920548163576284,
 0.9252262956748515,
 1.9192083413443877,
 1.8519357290245482,
 0.5407092054221374,
 1.7262865478623317,
 1.6647803569931081,
 1.3521530691248411,
 1.423595094849263,
 1.6106994068043785,
 2.683731394102202,
 3.2004466725467298,
 2.7311136604783943,
 2.349444116800975,
 0.8017560590798836,
 2.263340885775492,
 1.2849579142144503,
 2.2285925632952117,
 1.6136292804662589,
 3.3466623416562813,
 2.727614223898756,
 3.097869728969683,
 1.8574773630475796,
 1.5975669740432124,
 0.8619224508068133,
 1.1104742572431077,
 1.450596106348239,
 1.3467850901038192,
 2.2513999833891125,
 2.8681088967202335,
 2.3447252026575867,
 1.6647803569931081,
 1.8098596252263643,
 2.3706781679537627,
 1.6647803569931081,
 1.8210586104296862,
 0.7019708665640318,
 1.2918868467971938,
 3.5536988963324396,
 2.0816312580474396,
 3.463769562507756,
 2.932304343842135,
 1.5004086384356983,
 1.7025626894786368,
 1.2388210063460579,
 1.1132842316044038,
 0.6587303588154946]
```

Is there a difference between our approach and the implementation in scipy?

```
[ ]: difference_mahalanobis = mahalanobis_distance - setosa_mahalanobis
     difference_mahalanobis
```

```
[ ]: array([ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  2.22044605e-16,
             0.00000000e+00, -2.22044605e-16,  0.00000000e+00,  0.00000000e+00,
         0.00000000e+00,  2.22044605e-16,  0.00000000e+00,  0.00000000e+00,
        -2.22044605e-16,  0.00000000e+00,  0.00000000e+00,  4.44089210e-16,
         0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
         0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
         0.00000000e+00,  0.00000000e+00,  2.22044605e-16,  0.00000000e+00,
         0.00000000e+00, -2.22044605e-16,  0.00000000e+00,  0.00000000e+00,
         0.00000000e+00,  0.00000000e+00,  2.22044605e-16,  0.00000000e+00,
         0.00000000e+00,  2.22044605e-16,  0.00000000e+00,  0.00000000e+00,
         0.00000000e+00,  0.00000000e+00,  0.00000000e+00, -4.44089210e-16,
         0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
         0.00000000e+00,  0.00000000e+00])
```

Within in numerical precision, there is no difference. So, we can conclude that we implemented mahalanobis distance appropriately!

Part 2

```
[ ]: min_malahanobis_setosa = np.min(setosa_mahalanobis)
     max_malahanobis_setosa = np.max(setosa_mahalanobis)

     range_malahanobis_setosa = max_malahanobis_setosa - min_malahanobis_setosa
     range_malahanobis_setosa
```

```
[ ]: 3.0129896909103024
```

What is the maximum for the whole dataset?

```
[ ]: max_malahanobis_setosa
```

```
[ ]: 3.5536988963324396
```

Also the minimum?

```
[ ]: min_malahanobis_setosa
```

```
[ ]: 0.5407092054221374
```

Part 3 Now we can remove the maximum value from the dataframe to calculate the new range

```
[ ]: setosa_remove_max_df = setosa_df.drop(index= np.argmax(max_malahanobis_setosa))
```

Recalculate the malahanobis distances based on this new dataset

```
[ ]: new_setosa_mahalanobis = create_mahalanobis_distance(setosa_remove_max_df.
       ↪values)
     new_setosa_mahalanobis
```

```
[ ]: array([1.38013599, 1.13623906, 1.27339791, 0.92489802, 1.89535927,
            1.82847778, 0.5390951 , 1.70408027, 1.65227722, 1.35109978,
        1.40495211, 1.60001728, 2.66003785, 3.18969141, 2.70609061,
        2.32813621, 0.79448608, 2.24206687, 1.27030293, 2.20729521,
        1.59218978, 3.31821403, 2.69689375, 3.06289963, 1.83643754,
        1.57498605, 0.86307787, 1.11171646, 1.43043985, 1.32803482,
        2.22632991, 2.84820855, 2.33533575, 1.65227722, 1.8014304 ,
        2.36158106, 1.65227722, 1.79962071, 0.70175128, 1.28074088,
        3.51460882, 2.05820636, 3.42710832, 2.89932221, 1.48042193,
        1.68644336, 1.22299441, 1.11398908, 0.66189184])
```

```
[ ]: new_max_malahanobis_setosa = np.max(new_setosa_mahalanobis)
     new_max_malahanobis_setosa
```

```
[ ]: 3.5146088187318156
```

Ok, what is the new range?

```
[ ]: new_min_malahanobis_setosa = np.min(new_setosa_mahalanobis)
     new_min_malahanobis_setosa
```

```
[ ]: 0.5390951013136559
```

```
[ ]: new_range_malahanobis_setosa = new_max_malahanobis_setosa -␣
       ↪new_min_malahanobis_setosa
     new_range_malahanobis_setosa
```

```
[ ]: 2.97551371741816
```

Yes, the range between the values have decreased once the maximum value has been removed. Here is the difference between the ranges:

```
[ ]: range_malahanobis_setosa - new_range_malahanobis_setosa
```

```
[ ]: 0.0374759734921426
```

Obviously, the max has decreased but the minimum has also changed:

```
[ ]: max_malahanobis_setosa - new_max_malahanobis_setosa
```

```
[ ]: 0.039090077600623996
```

```
[ ]: min_malahanobis_setosa - new_min_malahanobis_setosa
```

```
[ ]: 0.0016141041084815066
```

Yes, the range of the distance do appear to be closer. The range has gotten smaller once the largest mahalanobis distance values were removed. Yes, the range is smaller once the largest mahalanobis distance is removed.

2.0.1 Part 4

Above, we first calculated the mahalanobis distance. Recall, that the mahalanobis distance is a metric that describes the distance from a point to a distribution. In our case, we were able to see the distance between our observation and the variance that occurs for each of the features. For the `setosa` species this means that an observation with a high malahanobis distance is less representative of the dataset as a whole. This is reinforced by looking at parts 2 and 3. We see that the range of decreases. Which means that extreme observations (max and min) are now more representative and the covariance has reduced

3 3. Module 4 Note this is not a Collaborative Problem

20 Points Total

In this problem we would like to introduce you to data transformations that can be used in order to generate features and extract components of images. For this problem you will be working with the wavelet transform. You may use the built-in function for this. Please follow the steps outlined below.

1. [2 points] Read in the MNIST dataset.
2. [2 points] Reshape the arrays in order to visualize each image in the dataset (in the MNIST dataset, each row represents a 28x28 image).
3. [2 points] Find an index from the MNIST dataset that holds an image for the number 7, we will use this in the subsequent steps.
4. [6 points] Perform a wavelet transform on this image and retain only the vertical and diagonal components of this image.
5. [6 points] Perform the inverse transform to revert back to an image.
6. [2 points] Plot the image (using plt.imshow) and provide an analysis of what was accomplished in this problem.

Part 1

```
[ ]: ## Type code here for part 1 ##
     mnist_df = pd.read_csv("train.csv")
     mnist_values = mnist_df.values
```

```
[ ]: mnist_df
```

```
[ ]:        label  pixel0  pixel1  pixel2  pixel3  pixel4  pixel5  pixel6  pixel7  \
     0          1       0       0       0       0       0       0       0       0
     1          0       0       0       0       0       0       0       0       0
     2          1       0       0       0       0       0       0       0       0
     3          4       0       0       0       0       0       0       0       0
     4          0       0       0       0       0       0       0       0       0
     …        …       …       …       …       …       …       …       …       …
     41995      0       0       0       0       0       0       0       0       0
     41996      1       0       0       0       0       0       0       0       0
     41997      7       0       0       0       0       0       0       0       0
     41998      6       0       0       0       0       0       0       0       0
```

```
41999      9       0       0       0       0       0       0       0       0

       pixel8  …  pixel774  pixel775  pixel776  pixel777  pixel778  \
0           0  …         0         0         0         0         0
1           0  …         0         0         0         0         0
2           0  …         0         0         0         0         0
3           0  …         0         0         0         0         0
4           0  …         0         0         0         0         0
…         …   …       …       …       …       …       …
41995       0  …         0         0         0         0         0
41996       0  …         0         0         0         0         0
41997       0  …         0         0         0         0         0
41998       0  …         0         0         0         0         0
41999       0  …         0         0         0         0         0

       pixel779  pixel780  pixel781  pixel782  pixel783
0             0         0         0         0         0
1             0         0         0         0         0
2             0         0         0         0         0
3             0         0         0         0         0
4             0         0         0         0         0
…           …       …       …       …       …
41995         0         0         0         0         0
41996         0         0         0         0         0
41997         0         0         0         0         0
41998         0         0         0         0         0
41999         0         0         0         0         0

[42000 rows x 785 columns]
```

The number displayed is the first value in each of the columns. Let's extract this for later to identify the number 7

Part 2

```
[ ]: mnist_labels = mnist_values[0][:]
```

Now, we can reshape each of these rows to get a 28 x 28 matrix for displaying

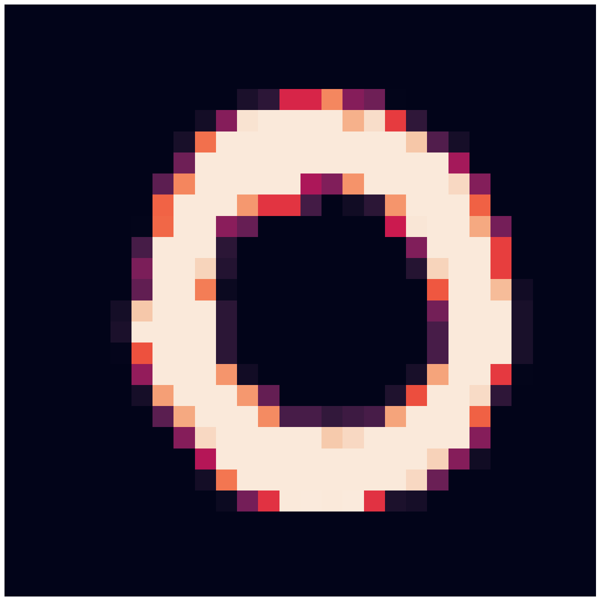
```
[ ]: mnist_numbers = [row[:784].reshape(28,28) for row in mnist_values]
```

Let's display one of these numbers to make sure we did this right!

```
[ ]: plt.figure(figsize=(12, 4))

     plt.imshow(mnist_numbers[1])
     plt.title("Raw image")
     plt.axis('off')
plt.show()
```

Raw image



We should see `0` in the second index of the `mnist_labels` list

```
[ ]: mnist_labels[1]
```

```
[ ]: 0
```

Cool! Looks like we did the pre-processing right

Part 3 We can find the number seven by using the label column in our dataframe and take the first element that appears in our list

```
[ ]: ## Type code here for part 3 ##
     seven_index = mnist_df.index[mnist_df["label"] == 7].tolist()[0]
     seven_index
```

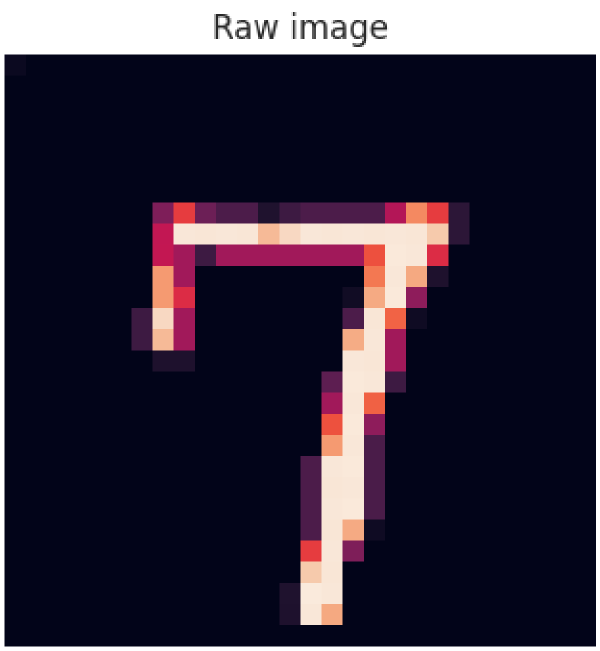
```
[ ]: 6
```

Let's plot the 6th index to make sure we get the number `7`

```
[ ]: plt.figure(figsize=(12, 4))

     plt.imshow(mnist_numbers[6])
     plt.title("Raw image")
     plt.axis('off')

     plt.show()
```



Now we have the number seven displayed!

```
[ ]: seven = mnist_numbers[6]
```

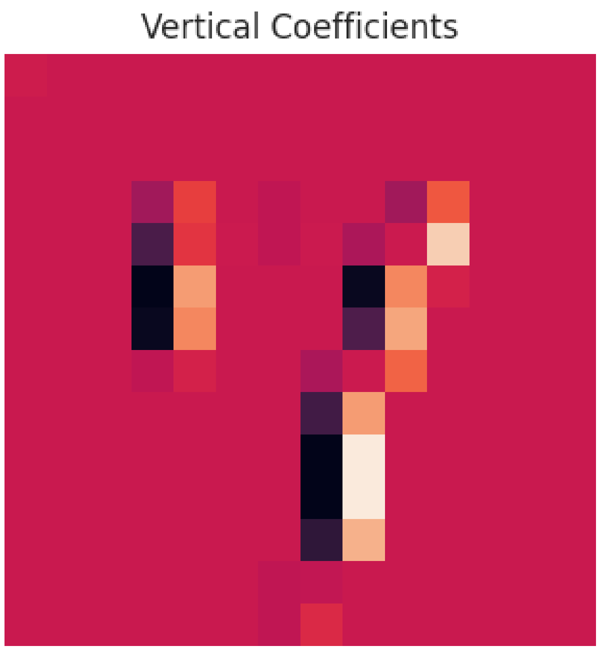
Part 4 We can perform the wavelet transform to see the reduction of the image

```
[ ]: coeff = pywt.dwt2(seven, "haar")
     _, (_, c_vertical, c_diagonal) = coeff
```

```
[ ]: plt.figure(figsize=(12, 4))

     plt.imshow(c_vertical)
     plt.title("Vertical Coefficients")
     plt.axis('off')

     plt.show()
```



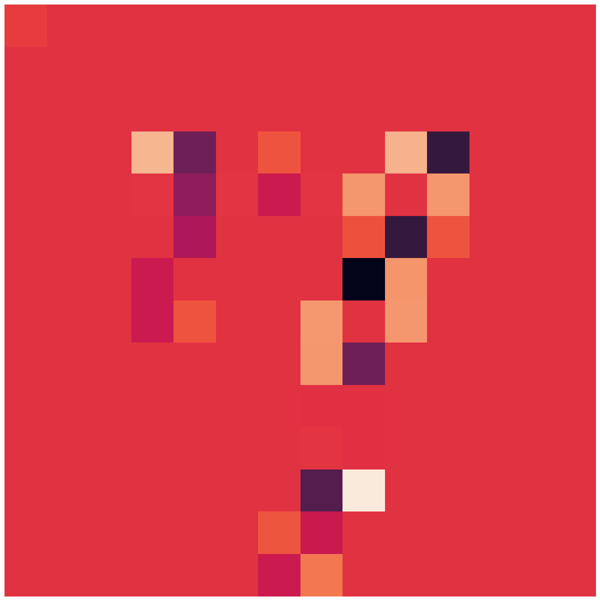
This plot helps to show the High-frequency components that show the average change in the vertical pixels

```
[ ]: plt.figure(figsize=(12, 4))

     plt.imshow(c_diagonal)
     plt.title("Diagonal Coefficients")
     plt.axis('off')

     plt.show()
```

Diagonal Coefficients



Similarly, the cofficients on the diagonal help to show the average differences between the pixels on the diagonal

```
[ ]: reduced_coeff = _, (_, c_vertical, c_diagonal)
```

```
[ ]: reduced_coeff[1][2]
```

```
[ ]: array([[ 3.50000000e+00,  0.00000000e+00,  0.00000000e+00,
              0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
              0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
              0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
              0.00000000e+00,  0.00000000e+00],
            [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
              0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
              0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
              0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
              0.00000000e+00,  0.00000000e+00],
            [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
              0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
              0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
              0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
              0.00000000e+00,  0.00000000e+00],
            [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
              4.10000000e+01, -4.05000000e+01,  0.00000000e+00,
              1.00000000e+01,  0.00000000e+00,  0.00000000e+00,
              4.00000000e+01, -6.10000000e+01,  0.00000000e+00,
              0.00000000e+00,  0.00000000e+00],
            [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
              5.00000000e-01, -3.00000000e+01,  5.00000000e-01,
             -1.00000000e+01,  5.00000000e-01,  3.05000000e+01,
              2.01505479e-14,  3.05000000e+01,  0.00000000e+00,
              0.00000000e+00,  0.00000000e+00],
            [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
              0.00000000e+00, -2.00000000e+01,  0.00000000e+00,
              0.00000000e+00,  0.00000000e+00,  9.50000000e+00,
             -6.10000000e+01,  1.00000000e+01,  0.00000000e+00,
              0.00000000e+00,  0.00000000e+00],
            [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
             -1.00000000e+01,  0.00000000e+00,  0.00000000e+00,
              0.00000000e+00,  0.00000000e+00, -8.10000000e+01,
              3.00000000e+01,  0.00000000e+00,  0.00000000e+00,
              0.00000000e+00,  0.00000000e+00],
            [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
             -1.00000000e+01,  1.00000000e+01,  0.00000000e+00,
              0.00000000e+00,  3.10000000e+01,  2.01505479e-14,
              3.05000000e+01,  0.00000000e+00,  0.00000000e+00,
              0.00000000e+00,  0.00000000e+00],
            [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
              0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
              0.00000000e+00,  3.05000000e+01, -4.05000000e+01,
              0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
              0.00000000e+00,  0.00000000e+00],
            [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
              0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
              0.00000000e+00, -5.00000000e-01, -5.00000000e-01,
              0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
              0.00000000e+00,  0.00000000e+00],
            [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
              0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
              0.00000000e+00,  5.00000000e-01, -5.00000000e-01,
              0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
              0.00000000e+00,  0.00000000e+00],
            [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
              0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
              0.00000000e+00, -5.00000000e+01,  6.05000000e+01,
              0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
              0.00000000e+00,  0.00000000e+00],
            [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
              0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
              1.05000000e+01, -1.05000000e+01,  0.00000000e+00,
              0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
              0.00000000e+00,  0.00000000e+00],
            [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
              0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
             -1.00000000e+01,  2.05000000e+01,  0.00000000e+00,
              0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
              0.00000000e+00,  0.00000000e+00]])
```

[ ]:
```
## Type code here for part 5 ##
reconstruct_seven = pywt.idwt2(reduced_coeff, "haar")
```

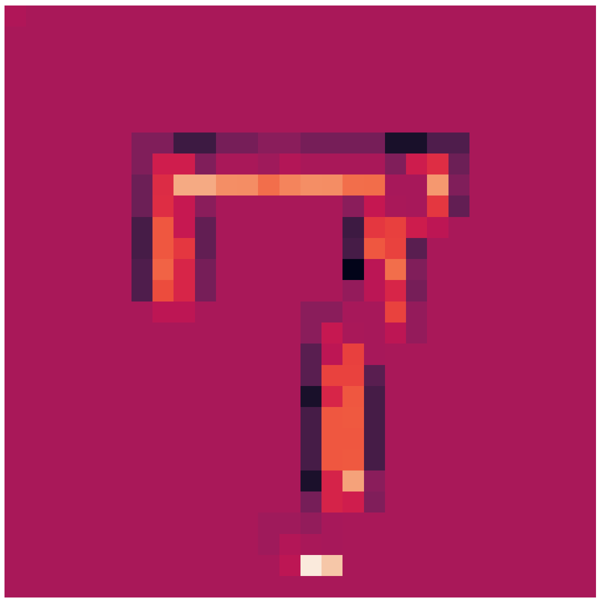
[ ]:
```
## Type code here for part 6 ##

plt.figure(figsize=(12, 4))

plt.imshow(reconstruct_seven)
plt.title("Reconstructed Image")
plt.axis('off')

plt.show()
```

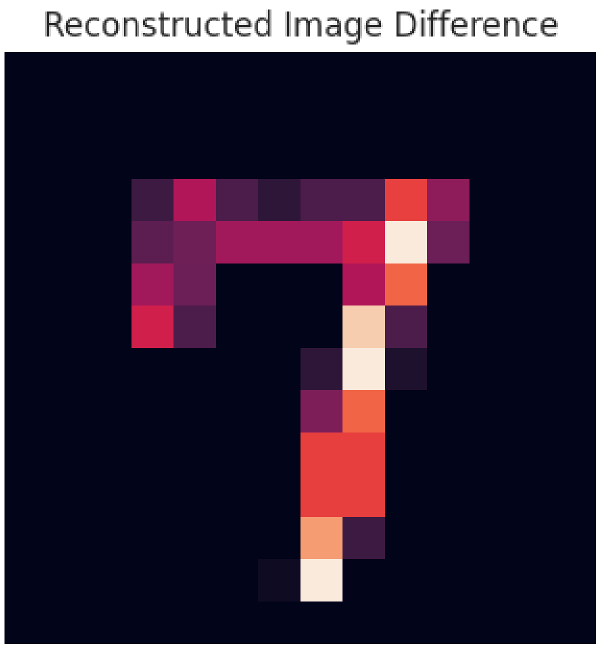
Reconstructed Image



[ ]:
```
plt.figure(figsize=(12, 4))

plt.imshow(seven - reconstruct_seven)
plt.title("Reconstructed Image Difference")
plt.axis('off')

plt.show()
```



The purpose of this exercise is to show how the wavelet transform and inverse wavelet transform can be used to extract the components of an image. By taking on the vertical and diagonal coefficient we can construct a lower fidelity representation of the image 7. However, we can see that the reconstructed image mostly holds its shape. But the difference between the original and reconstructed image tells us that there is some changes between the transformation, lossy transform

4 4. Module 5 Note this is a Collaborative Problem

20 Points Total

In this problem we would like to familiarize you with a basic example of a goal based agent. A game like rock paper scissors is typically played between two players and has a clearly defined set of rules for a win, draw, and loss. Since it is impossible (acknowledging that studies have been preformed to use behavior to attempt a prediction - but we will ignore that for the purpose of this exercise) to predict an opponent's move, we will assume that the user will input a value, the computer will see this input then choose from a set of rules the correct response play, acting as a simple goal based agent.

Instructions on how to play: https://www.wikihow.com/Play-Rock,-Paper,-Scissors

For this problem, you will fill in the rules (in if-then-else format) to the function in the cell below. The proceeding cell should then be run to play a game with the computer in which the computer agent should win every time.

[ ]:
```
## Set Rock = 1, Paper = 2, Scissors = 3
## The funtion should read in the player_move and return the appropriate␣
 ↪agent_play that will win the round
```

```python
import random
def goal_based_agent(player_move=None):
    rock = 1
    paper = 2
    scissors = 3
    if player_move == rock:
        return paper
    if player_move == paper:
        return scissors
    if player_move == scissors:
        return rock
```

[ ]:
```python
## Set Rock = 1, Paper = 2, Scissors = 3
## The program will receive player input for the number of rounds to be played:␣
↪suggested to use 5 so that you may test your implementation
import sys

num_rounds = int(input("Enter number of rounds to be played: "))
curr_round = 1
win_counter = 0
while curr_round <= num_rounds:
    print("Enter choice \n 1 for Rock, \n 2 for paper, and \n 3 for scissor \n")

    # take the input from user
    player_move = int(input("User turn: "))

    if player_move == 1 or player_move == 2 or player_move == 3:
        if player_move == 1:
            choice_name = 'Rock'
        elif player_move == 2:
            choice_name = 'Paper'
        elif player_move == 3:
            choice_name = 'Scissor'

        print("Player move is: ", choice_name)

        AI_move = goal_based_agent(player_move) ## this is where your function␣
↪is called, player_move which is an input is sent to your agent and returns␣
↪the correct AI move

        if AI_move == 1:
            AI_play = 'Rock'
        elif AI_move == 2:
            AI_play = 'Paper'
        elif AI_move == 3:
            AI_play = 'Scissor'
        print("AI move is: ", AI_play)

        win = None
        if player_move == 1 and AI_move == 2:
            win = 1
        elif player_move == 2 and AI_move == 3:
            win = 1
        elif player_move == 3 and AI_move == 1:
            win = 1
        else:
            win = 0

        if win == 1:
            win_counter = win_counter + 1
            print("AI wins this round!")
        else:
            print("AI lost this round!")

        print("Round Number: ", curr_round)

        curr_round = curr_round + 1

    else:
        print("Invalid player input, GAME OVER")
        sys.exit(1)

print("\n\nGame Concluded")
print("Round Number: ", curr_round - 1)
print("Number of Wins: ", win_counter)
print("Number of Losses: ", curr_round - win_counter - 1)

if curr_round - win_counter == 1:
    print('Well done!')
else:
    print('You need to fix your goal based agnet!')
```

```
Enter choice
 1 for Rock,
 2 for paper, and
 3 for scissor

Player move is:  Rock
AI move is:  Paper
AI wins this round!
Round Number:  1
Enter choice
```

```
 1 for Rock,
 2 for paper, and
 3 for scissor

Player move is:  Paper
AI move is:  Scissor
AI wins this round!
Round Number:  2
Enter choice
 1 for Rock,
 2 for paper, and
 3 for scissor

Player move is:  Scissor
AI move is:  Rock
AI wins this round!
Round Number:  3
Enter choice
 1 for Rock,
 2 for paper, and
 3 for scissor

Player move is:  Rock
AI move is:  Paper
AI wins this round!
Round Number:  4
Enter choice
 1 for Rock,
 2 for paper, and
 3 for scissor

Player move is:  Paper
AI move is:  Scissor
AI wins this round!
Round Number:  5


Game Concluded
Round Number:  5
Number of Wins:  5
Number of Losses:  0
Well done!
```

5 References

[1] Bishop, Christopher M., Pattern Recognition and Machine Learning, Springer, 2006,
[2] Dillon, and Goldstein, M.. Multivariate Analysis Methods and Applications, John

Wiley, 1984 https://www.microsoft.com/en-us/research/uploads/prod/2006/01/Bishop-Pattern-Recognition- and-Machine-Learning-2006.pdf [3] Duin, Robert P.W., Tax, David and Pekalska, Elzbieta, PRTools, http://prtools.tudelft.nl/ [4] Fisher, R.A., The use of Multiple Measurements in Taxonomic Problems, Annals of Human Genetics, Vol. 7, Issue 2, pp. 179-188, 1936 [5] Hotelling, H., Analysis of a complex of statistical variables into principal components, Jour- nal of Educational Psychology, Number 24, pp. 417–441, 1933 [6] Rao, K. P. and Yip, P., Discrete Cosine Transform Algorithms, Advantages, Applications, San Diego, CA: Academic Press, Inc., 1990