The Coin Machine

Brazos Fitch and Yves Ngenzi

Presentation Outline

- Class Summary
- Problem Statement
- Original Plan, Encountered Issues, Final Product
  - Software/Code Design
  - Categories
  - Features
  - Training Data
- Lessons Learned/Conclusions
- References

Summary of PHYS453 - Pattern Recognition

Throughout the semester we learned about data analysis using python. We learned about classification and data manipulation. Hence, in this project, we put everything we learned from class to solve real world problem. Therefore, in this presentation, one expect to see application of python in data analysis.

Project Problem Statement

1. The development of an automated coin classifier and counter would largely decrease the time needed to retrieve an accurate value of the total currency amount of coins inside a jar.

2. The Goal of this project is to create a python script which would allow a user to upload an image of a random assortment of coins and then return a total value. This solutions could then replace coin counting machines in banks and kiosks.

Project Problem Statement

= $15.69

Software and Code Design - Original Plan

1. Object Recognition - masking
    a. Here we use software such as OpenCV for object recognition, and to crop and copy an image of a single coin
2. Image Processing - segmentation
    a. Resize the image and perform other processes to achieve best possible results. (i.e. rotation, color scale, contrast)
3. Classification
    a. Then use a classifier to classify the coin in the new image

Classification Categories - Original Plan

These are the classes we expect to see:

- Quarter
- Dime
- Nickel
- Penny
- Unknown

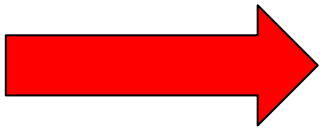
The Unknown category is intended to collect all objects that are not a U.S. coin

Which Features? - Original Plan

The most obvious features:

- Size (diameter)
- Color (shade)
- Pixels (in case of an image)

One classifier could use size and color whilst another compares images



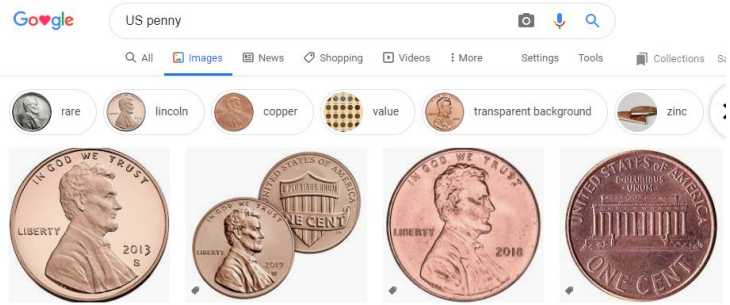
This of course would require new categories for the heads and tails sides of the coins

Training data from coin images - Original Plan

Training data can be made from using photos of individual coins found around the house or even exchanged at a bank

Training data (images) can even be found on Google and could undergo image preprocessing steps before being used to train the classifier

A large dataset can be made with the y dimension being items and the x dimension being features and coin type



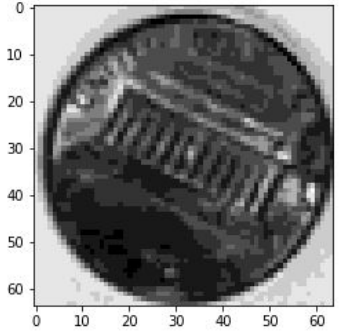
Categories - Encountered Issues

- Originally we were going to include an unknown category for objects which were not U.S. coins, but had a difficult enough time with just coins and decided to leave them out.

Features - Encountered Issues

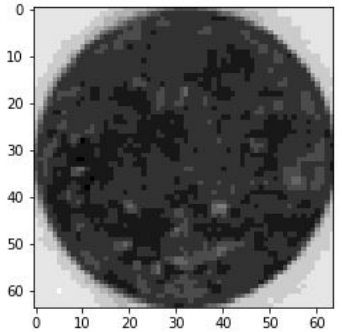
1. Process
    a. Finding the circles using Hough Circles
    b. Cropping into individual images
    c. Pixelating the images and looking at each individual pixel
2. Issues
    a. This would require each image to be reoriented to face the same direction
    b. Nearly impossible since each coin is a circle

```
[88] X_test = coincutter("attemptcoins1.jpg")
     test = X_test[0].reshape((64,64))
     plt.imshow(test,cmap = plt.cm.binary)
     plt.show()
```




```
[89] X_test = coincutter("attemptcoins2.jpg")
```

```
[90] test = X_test[0].reshape((64,64))
     plt.imshow(test,cmap = plt.cm.binary)
     plt.show()
```



Features - Encountered Issues

1. Hough Circle Transform
    a. OpenCV feature which locates circles in an image and extracts their locations and radii
    b. Parameters
        i. MinDist - minimum distance between circles
        ii. Min and Max Radius
        iii. Method specific parameters - used to fine tune how many circles it will find
2. Image Normalization
    a. It was difficult to have images that were similar enough for comparison
    b. Size, color, lighting, angle, etc.
3. Switch to Color and Radius

Training Data - Encountered Issues

- Have different size
- Different people took pictures differently
- They are oriente diffentry and all of these will affect our analysis

Categories - Final Product

1. Our Final Categories were just a number representing the relative size of the coins.
   a. Dime: 1
   b. Penny: 2
   c. Nickel: 3
   d. Quarter: 4
2. If we had represented them by their values, then it would have been easy to sum the results from the classifier and get a total.

```
[4] tst2 = coincutter("24dimes.jpg",1)
```

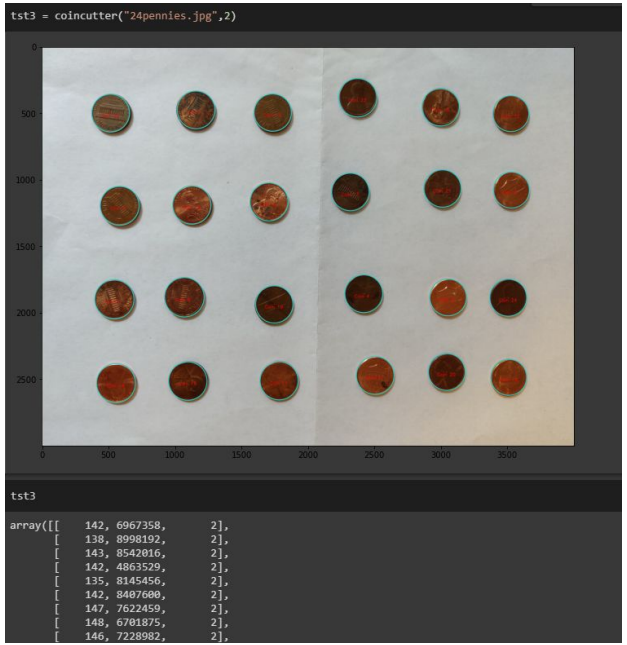

```
[5] tst2

array([[     120, 11182487,        1],
       [     123, 10919573,        1],
       [     118, 12039087,        1],
       [     120,  7892066,        1],
       [     121,  8024938,        1],
       [     117, 12431777,        1],
       [     124,  7827306,        1],
       [     123, 13287089,        1],
       [     123,  9341058,        1],
       [     123,  9076334,        1],
       [     128,  4603963,        1]
```

Features Final Product

1. Radius
    a. This was relatively simple to extract, assuming that Hough Circles worked accurately. Once that part had been complete, the radius was used to crop out the images for color extraction
2. Color
    a. Most of the times people look for a specific color, not ask what color something is.
    b. RGB value was easy to retrieve, which was then translated to hexadecimal and then again to decimal.
    c. There is a better way to train based on color.

Training and Test Data - Final Product

1. Use coincutter() to find all coins in an image and then extract radius, and color, and populate a numpy array with this information. Has option to include which coins they are for batch training data.



Training and Test Data - Final Product

Dimes

Pennies

Nickels

Quarters

Training and Test Data - Final Product

1. getmoney()
2. Finds individual coins on the test images and returns positions and radius for classifier.
3. This code is using a decision tree classifier from scklearn

```
def getmoney(image):
  X_coin = coincutter(image,0)
  X_coin = X_coin[:,[0,1]]
  results = clf3.predict(X_coin)
  amount = 0
  for i in range(0,results.shape[0]):
    if(results[i] == 4):
      amount += 0.25
    if(results[i] == 3):
      amount += 0.05
    if(results[i] == 2):
      amount += 0.01
    if(results[i] == 1):
      amount += 0.10
  print(amount)
  print(X_coin,results)
```

```
[ ] getmoney("classifythis1.jpg")

0.97
[[    163 4603957]
 [    162 7432013]
 [    138 6833460]
 [    152 6839887]
 [    135 8934970]
 [    130 7299657]
 [    133 4206633]
 [    128 5918528]
 [    124 8484448]] [4 4 2 3 2 1 1 1 1]
```

Google Image preparation

First steps

- Picked the coin images that were single coins
- Renamed them
- Uploaded them to google drive
- Uploaded these image to google colab
- Converted them into a list

Second steps

- Change the size of the image so they can all be the same
- Blurred images to avoid failure of the circle drawing around the coins
- Found the radius of the coins
- Used radius as my training data
- And I will use the same thing to the testing data

Image preparations

Training data

```
print('quarters',len(quarters))
print('nickels',len(Nickles))
print('Dime',len(Dimes))
print('Pennies',len(Pennies))
```

```
quarters 90
nickels 120
Dime 97
Pennies 147
```

Testing data

```
[254] print('quarters',len(quarters))
      print('nickels',len(Nickles))
      print('Dime',len(Dimes))
      print('Pennies',len(Pennies))
```

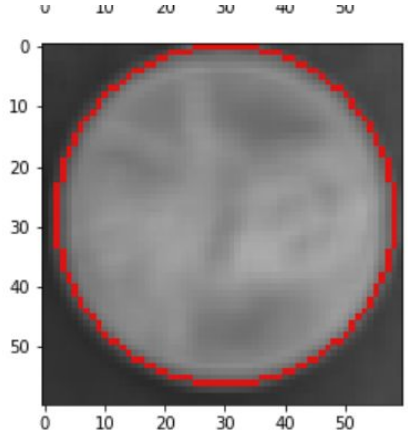
```
quarters 13
nickels 4
Dime 5
Pennies 25
```

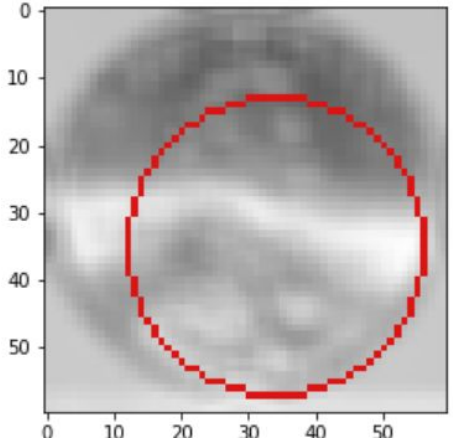
Results

```
clf.score(V,a)
```

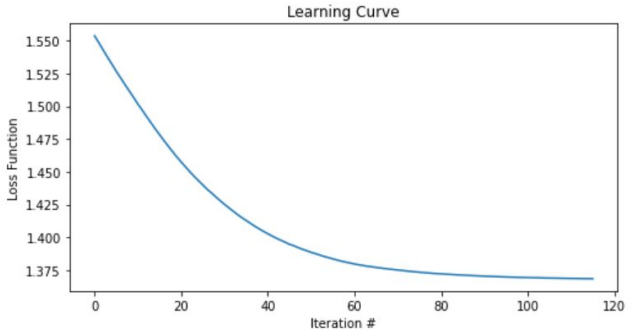
```
0.3237885462555066
```

Neural Network was scored 32%





```
[67] plt.figure()
     plt.plot(clf.loss_curve_)
     plt.title('Learning Curve')
     plt.xlabel('Iteration #')
     plt.ylabel('Loss Function')
     plt.show()
```



Testing Data

```
The number of quarters are: 0
The number of Pennies are: 47
The number of Nickles are: 0
The number of Dimes are: 0
```

Result

Bayes classification

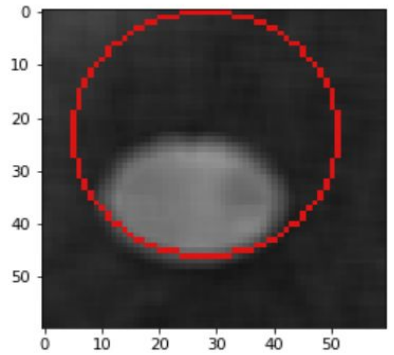
Did not do well but different for the same reason of bad radius recognition

```
print( "The number of quarters are:", len(a))

print( "The number of Pennies  are:", len(b))

print( "The number of Nickles are:", len(c))

print( "The number of Dimes are:", len(e))
```

```
The number of quarters are: 0
The number of Pennies  are: 41
The number of Nickles are: 6
The number of Dimes are: 0
```

```
[254] print('quarters',len(quarters))
      print('nickels',len(Nickles))
      print('Dime',len(Dimes))
      print('Pennies',len(Pennies))
```

```
quarters 13
nickels 4
Dime 5
Pennies 25
```



Results

K Nearest Neighbor

```
print( "The number of quarters are:", len(u))

print( "The number of Pennies are:", len(v))

print( "The number of Nickles are:", len(w))

print( "The number of Dimes are:", len(m))
```

```
The number of quarters are: 2
The number of Pennies are: 10
The number of Nickles are: 1
The number of Dimes are: 34
```

Testing data

```
[254] print('quarters',len(quarters))
      print('nickels',len(Nickles))
      print('Dime',len(Dimes))
      print('Pennies',len(Pennies))
```

```
quarters 13
nickels 4
Dime 5
Pennies 25
```

Results

Decision Tree

```
[ ]  from sklearn.tree import DecisionTreeClassifier
     import sklearn.tree

     clf3 = DecisionTreeClassifier(max_depth = 3)
     clf3.fit(X,y)
     clf3.score(X,y)

     0.9361702127659575
```

Coins: `[4 4 2 3 2 1 1 1 1]`

Total: `0.97`

Actual Total: **0.88**

Conclusions

1. Image Processing/Normalization
2. Time Management
3. Pattern Recognition
4. Using References
5. Working with Images
6. Building training data

GitHub: https://github.com/brazosfitch/Pattern-Recognition-Final-Project