ARTIFICIAL INTELLEGNECE

Lab 06

Abstract

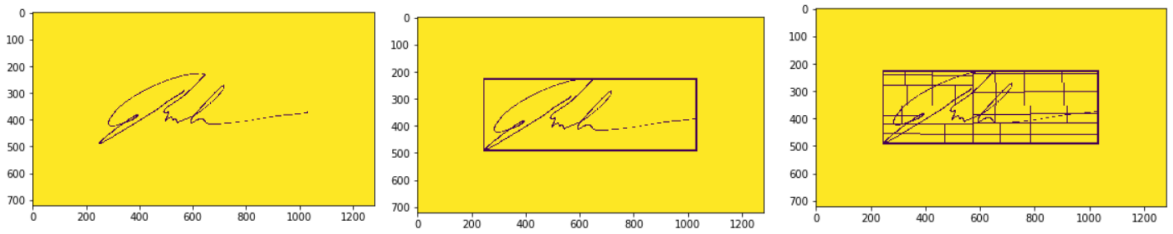
Final Report for Feature Extraction in Signature Verification

Syed Muhammad Akbar

Msyed.bscs17seecs

1 INTRODUCTION

Signature verification is still one of the most active field with state-of-the-art research continously deriving the field forward. However, in order to truly appreciate the progress in this field, it is essential that we come about after going over the fundamentle. The earliest progress in this field was driven by comparing and evaluating the differences in the features between the new and original signatures. However, in order to compare the features it was necessary that we have the features. To achieve this task several techniques were used to derive independent features with the process known as Feature Extraction.



2 FEATURE EXTRACTION

To begin with, we were assigned task to find the centroid and divide the signature into four segments. To achieve this, first of all, I created a bounding box enclosing the entire signature and saved the image with co-ordinates of the bounding box in a list. Then in a function named centroid(), I used to the co-ordinates of this bounding box find out the centroid. The centroid value and the list was passed in the function div_four_dim() which returned the bounding box divided into four segments. The previous methodoly was recursively repeated twice to obtain 64 individual segments.

2.1 BLACK TO WHITE TRANSITIONS

Black to white transition is the transition when the white pixel comes after the black pixel. This was simply calculated by checking if the current pixel is white and the previous pixel is black. All such transitions were summed which was in terms returned by the function.

```
def white_transition(img):
  height, width = img.shape
  prev = img[0, 0]
  n = 0
  for x in range(1, width):
    for y in range(1, height):
      curr = img[y, x]
      if curr > 200 and prev < 50:
        n = n + 1
      prev = curr
  #print(n)
  return (n)
```

2.2 ASPECT RATIO (= WIDTH/HEIGHT)

It was the simplest where we divided the width by height.

```
def aspect_ratio(seg):
  n = (seg[2] - seg[4]) / (seg[1] - seg[3])
  return (n)
```

2.3 NUMBER OF BLACK PIXELS

This was achieved by counting the number of black pixels in the segment.

```
def num_of_black(img):
  height, width = img.shape
  prev = img[0, 0]
  n = 0
  for x in range(1, width):
    for y in range(1, height):
      curr = img[y, x]
      if curr < 50:
        n = n + 1
  #print(n)
  return (n)

i = 0
black_list = []
for segment in segments:
  for seg in segment:
    i = i + 1
    black_list = black_list + [num_of_black(seg)]
    print('Segment # ', i, ": # black_pixelds", num_of_black(seg))
```

2.4 NORMALIZED SIZE

In this method of feature extraction we calculated the size of the segment and divided it the total number of black pixels.

```
s_list = []
#black_list
#seg_16by16
i = 0
for segments in seg_16by16:
  for seg in segments:
    temp = (seg[1] - seg[3]) * (seg[2] - seg[4]) / black_list[i]
    i = i + 1
    s_list = s_list + [temp]
    print('Segment # ', i, "\t\t normalized size = ", s_list[i-1])
```

2.5 Inclination of Centroid

We calculated the angle of the line from centroid to bottom-left corner.

```
import math
c_angle_list = []
#black_list
#seg_16by16
i = 0
for segments in seg_16by16:
  for seg in segments:
    temp_cent = centroid(seg)
    temp = math.atan((seg[3] - temp_cent[0])/(temp_cent[1] - seg[4]))
    i = i + 1
    c_angle_list = c_angle_list + [temp]
    print('Segment # ', i, "\t\t inclination of centroid = ", c_angle_list[i-1])
```

2.6 Normalized inclination of black-pixels

The previous process was repeated but for all the black pixels instead. All the answers were summed up and divided by the total number of black pixels.

```
def normalized_inclination(img):
  height, width = img[0].shape
  #prev = img[0, 0]
  n = 0
  temp_ret = 0
  for x in range(1, width):
    for y in range(1, height):
      curr = img[0][y, x]
      if curr < 50:
        #temp_cent = centroid(seg)
        try:
          temp = math.atan((img[4] - x)/( y - img[3]))
          temp = math.fabs(temp)
          temp_ret += temp
          n += 1
        except:
          pass
  #print(n)
  #print('Segment # ', i, "\t\t normalized_inclination = ", n_inc[i-1])
  n = temp_ret / n
  return (n)

n_inc = []
i = 0
for segments in seg_16by16:
  for seg in segments:
    #temp = (seg[1] - seg[3]) * (seg[2] - seg[4]) / black_list[i]
    i = i + 1
    n_inc = n_inc + [normalized_inclination(seg)]
    print('Segment # ', i, "\t\t normalized_inclination = ", n_inc[i-1])
```

3 TOP 3 FEATURES

The rational behind my top three features is that they all need to be independent among themselves because dependent features ends up being a constant factor of other and causing problem in machine learning. Therefore the following are my top three features:

1) Normalized inclination of black-pixel
2) Black to white transitions in cell
3) Aspect ratio of cell

4 CONCLUSION

This was a very learning experience for me who wanted to learn about the premitive machine learning methodologies and about the origins modern machine learning. During the history, it was always believed that humans uses feature for identification and classification which is proven not to be entirely ture now. However, it is fascinating that how were we still able to use to progress in this field. Although a little less relevent today due to DNN and vast amount of data, feature extraction reducing the need of deeper layers and decreases the amount of data even today. Moreover, these lab were also a good introduction to python, numpy and other widely used libraries which will help progress in the field of machine learning.