CSE 151

ZEQUN YU

PA 4

Q1.

|  |  |  |  |
| --- | --- | --- | --- |
|  | P=3 | P=4 | P=5 |
| Training error | 0.0124 | 0.0069 | 0.0069 |
| Testing error | 0.0409 | 0.0264 | 0.0343 |

Q2.

|  |  |  |  |
| --- | --- | --- | --- |
|  | P=3 | P=4 | P=5 |
| Training error | 0.0127 | 0.0074 | 0.0069 |
| Testing error | 0.0541 | 0.0290 | 0.0340 |

Q3.

The two substrings corresponding to coordinates:

KVGPD

WDTAG

*# Author: Zequn Yu  
# iD: A14712777***from** \_\_future\_\_ **import** division  
**import** numpy  
**from** collections **import** defaultdict  
**from** collections **import** Counter  
  
*#'''  
# read data from train files*train\_file = open(**"pa4train.txt"**, **"r"**)  
train\_data = []  
*# set local var*blank = **' '***# for each line in fle***for** line **in** train\_file:  
 *# split by blank* ASCII\_LABEL = line.split(blank)  
 *# ascii contains ABCDEF...  
 # label contains* ascii = ASCII\_LABEL[0]  
 label = ASCII\_LABEL[1]  
 *# check and convert string to integer* **if** label == **"+1"**:  
 label = 1  
 **else** :  
 label = -1  
 *# add to data set* train\_data.append([ascii,label])  
  
*# read data from test file*test\_file = open(**"pa4test.txt"**, **"r"**)  
test\_data = []  
**for** line **in** test\_file:  
 T\_ASCII\_LABEL = line.split(blank)  
 *# read data* T\_ASCII = T\_ASCII\_LABEL[0]  
 T\_LABEL = T\_ASCII\_LABEL[1]  
 *# convert string to integer* **if** T\_LABEL == **'+1'**:  
 T\_LABEL = 1  
 **else** :  
 T\_LABEL = -1  
 *# add to data set* test\_data.append([T\_ASCII, T\_LABEL])  
*#'''***''' question 1 '''***# use the string kernel function  
# two strings s and t, the string kernel Kp(s, t) is the number of substrings of length p  
#'''***def** percepie(p):  
 per\_return = []  
 idx = 0  
 **for** data\_sec **in** train\_data:  
 num = 0  
 **for** prev **in** per\_return:  
 num += update\_num(data\_sec, prev, train\_data, p)  
 **if** num <= 0:  
 per\_return.append(idx)  
 idx = idx + 1  
 **return** per\_return  
  
  
**def** update\_num(data, prv, td, p):  
 ret = data[1] \* td[prv][1] \* string\_kernel(data[0], td[prv][0], p)  
 **return** ret  
  
**def** string\_kernel(s, t, p):  
 count = 0  
 s\_dic = defaultdict(int)  
 t\_dic = defaultdict(int)  
 *# loop* **for** ind\_s **in** range(len(s) - p + 1):  
 s\_idx = s[ind\_s:ind\_s + p]  
 s\_dic[s\_idx] += 1  
 **for** ind\_t **in** range(len(t) - p + 1):  
 t\_idx = t[ind\_t:ind\_t + p]  
 t\_dic[t\_idx] += 1  
  
 **for** k, v **in** s\_dic.items():  
 **if** k **in** t\_dic:  
 count += v \* t\_dic[k]  
 **return** count  
  
*# get test stat***def** test\_perceptron(t, s, p):  
 correct = 0  
 **for** data **in** t:  
 pred = 0  
 num = 0  
 **for** prev **in** s:  
 num += update\_num\_t(train\_data, prev, data, p)  
 **if** num <= 0:  
 pred = -1  
 **else**:  
 pred = 1  
 **if** pred == data[1]:  
 correct += 1  
 **return** correct  
  
**def** update\_num\_t(td, prev, data, p):  
 ret = td[prev][1] \* string\_kernel(data[0], td[prev][0], p)  
  
*# calculate errors  
# for 3, 4, 5***for** p **in** range(3,6):  
*#for p in range(2, 3):  
 #print("test p is:", p)* t = percepie(p)  
 s = train\_data  
 train\_cor = test\_perceptron(s, t, p)  
 x = (len(train\_data)-train\_cor)/len(train\_data)  
 print(**"When p is: "**, p, **" training error is: "**, (**'{0:.4f}'**.format(x)))  
 s = test\_data  
 test\_cor = test\_perceptron(s, t, p)  
 y = (len(test\_data)-test\_cor)/len(test\_data)  
 print(**"When p is: "**, p, **" test error is: "**, (**'{0:.4f}'**.format(y)))  
*#'''***''' question 2 '''***#'''***def** p2(s, t, p):  
 count = 0  
 s\_dict = defaultdict(int)  
 t\_dict = defaultdict(int)  
 **for** ind\_s **in** range(len(s) - p + 1):  
 s\_idx = s[ind\_s:ind\_s + p]  
 s\_dict[s\_idx] += 1  
 **for** ind\_t **in** range(len(t) - p + 1):  
 t\_idx = t[ind\_t:ind\_t + p]  
 t\_dict[t\_idx] += 1  
 **for** k, v **in** s\_dict.items():  
 **if** k **in** t\_dict:  
 count += 1  
 **return** count  
  
**def** percepie2(p):  
 res = []  
 idx = 0  
 **for** data\_sec **in** train\_data:  
 num = 0  
 **for** prev **in** res:  
 num += update\_num\_m(data\_sec, train\_data, prev, p)  
 **if** num <= 0:  
 res.append(idx)  
 idx = idx + 1  
 **return** res  
  
**def** update\_num\_m(data, td, prev, p):  
 res = data[1] \* td[prev][1] \* p2(data[0], td[prev][0], p)  
 **return** res  
  
**def** test\_perceptron2(t, s, p):  
 preds = []  
 correct = 0  
 **for** data\_sec **in** t:  
 pred = 0  
 num = 0  
 **for** prev **in** s:  
 num += update\_num\_m2(data\_sec, prev, p, train\_data)  
 **if** num <= 0:  
 pred = -1  
 **else**:  
 pred = 1  
 **if** pred == data\_sec[1]:  
 correct += 1  
 **return** correct  
  
**def** update\_num\_m2(data, prev, p, td):  
 res = data[prev][1] \* p2(data[0], td[prev][0], p)  
 **return** res  
  
**for** p **in** range(3,6):  
 w = percepie2(p)  
 train\_cor = test\_perceptron2(train\_data, w, p)  
 x = (len(train\_data)-train\_cor)/len(train\_data)  
 print(**"When p is: "**, p, **" training error is: "**, (**'{0:.4f}'**.format(x)))  
 test\_cor = test\_perceptron2(test\_data, w, p)  
 y = (len(test\_data) - test\_cor) / len(test\_data)  
 print(**"When p is: "**, p, **" testing error is: "**, (**'{0:.4f}'**.format(y)))  
  
*#'''***''' question 3 '''***#'''***def most\_cor**(s, p):  
 substrings = Counter()  
  
 **for** prev **in** s:  
 strng = train\_data[prev][0]  
 **for** idx\_s **in** range(len(strng) - p + 1):

s\_idx = [idx\_s:idx\_s+p]  
 substrings[strng[s\_idx]] += train\_data[prev][1]  
  
 **return** substrings.most\_common(2)  
  
w = percepie(5)  
  
x = most\_cor(w, 5)  
  
print(x)  
*#'''*