Homework 2

Yihui He

I. Problem 1

(a) $1 + 8 = 9,\ x_{13}, x_{15}$
(b) $\frac{1+8}{20(20-12)} = 0.05625$
(c) $1 - ErrRate = 0.94375$
(d) Shown in figure 1

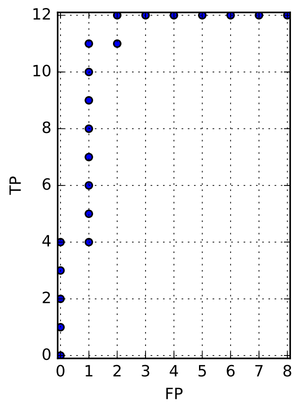

Fig. 1. problem 1(d)

II. Problem 2

(a) Shown in figure 7

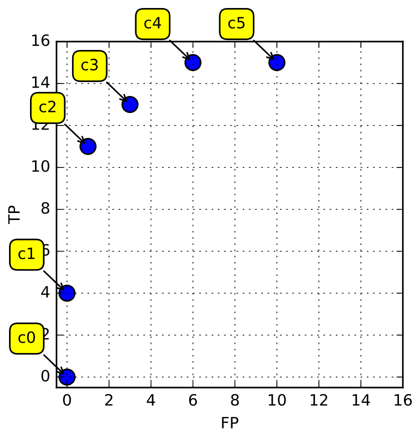

Fig. 2. problem 2(a)

(b) Shown in figure 3

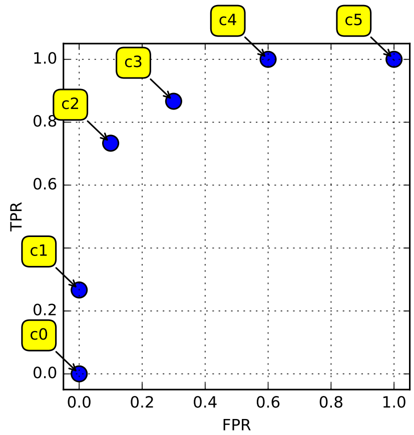

Fig. 3. problem 2(b)

(c) C_0 have the lowest accuracy. C_2, C_3 both have the highest accuracy.

$$accuracy = \frac{TP + N - FP}{N + P} \tag{1}$$

All accuracy shown below:

c0	0.4
c1	0.56
c2	0.8
c3	0.8
c4	0.76
c5	0.6

(d) C_2 have the highest average recall, C_0, C_5have the lowest average recall

$$avg - rec = (tpr + tnr)/2 \tag{2}$$

All avg-rec shown below:

c0	0.5
c1	0.63
c2	0.82
c3	0.78
c4	0.7
c5	0.5

(e) C_5 and C_4 are complete. A complete classifier has all positive examples in positive prediction.
(f) C_0 and C_1 are consistent. A consistent classifier has no negative example in positive prediction.

III. PROBLEM 3

Note that, margin caculation is below(no ——w—— normlization is common pitfall). Fomulas of loss are from textbook.

$$margin = c(X)\frac{W \cdot X - t}{\|W\|} \tag{3}$$

Code is as follow:

```
w=np.array([2,1,3])
t=12
inp=np.array(
[[2, 2, 3 ],[3, 3, 2 ],[1, 2, 3 ],[1, 4, 1 ],
[4, 4, 4 ],[2, 2, 2 ],[1, 1, 1 ],[0, 4, 2 ],
[4, 0, 0 ],[3, 3, 1 ],[3, 3, 3 ]])
label=np.array(
[1 if i <6 else -1 for i in range(len(inp))])
margin=label*(inp.dot(w.T)-t)/np.linalg.norm(w)
(margin <0).astype(int)#0-1
(np.maximum(1-margin ,0)#hinge
(1-margin)**2#square
```

	$margin$	$0-1$	$hinge$	$square$
0	0.801784	0.0	0.198216	0.039290
1	0.801784	0.0	0.198216	0.039290
2	0.267261	0.0	0.732739	0.536906
3	−0.801784	1.0	1.801784	3.246425
4	3.207135	0.0	0.000000	4.871444
5	0.000000	0.0	1.000000	1.000000
6	1.603567	0.0	0.000000	0.364294
7	0.534522	0.0	0.465478	0.216669
8	1.069045	0.0	0.000000	0.004767
9	0.000000	0.0	1.000000	1.000000
10	−1.603567	1.0	2.603567	6.778563

IV. PROBLEM 4

	$relative$	$laplace$	$1-estimate$	$1-estimat$
1	0.08	0.100000	0.100000	0.133333
2	0.16	0.166667	0.166667	0.177778
3	0.32	0.300000	0.300000	0.266667
4	0.00	0.033333	0.033333	0.088889
5	0.44	0.400000	0.400000	0.333333

My code of computing probability and generating figures is as follow:

```
inp=pd.Series(
[3,5,5,2,3,1,5,3,5,5,2,
3,5,5,1,2,3,3,5,3,3,5,5,2,5])
result=[]
relative=inp.value_counts(
  normalize=True, sort=False)
  .reindex(range(1,6),fill_value=0)
result.append(relative)
nominator=
  (inp.value_counts()
  .reindex(range(1,6),fill_value=0)+1)
laplace=nominator/sum(nominator)
result.append(laplace)
result.append(laplace)
nominator=(
inp.value_counts().
  reindex(range(1,6),fill_value=0)+4)
m_est=nominator/sum(nominator)
result.append(m_est)
result=pd.DataFrame(
result,index=
  ['relative','laplace','1-est','5-est']).T
for i in result:
    result[i].plot()
    plt.xticks(range(1,6))
    plt.title(i)
    plt.grid(True)
    plt.savefig(i+'.eps')
    plt.show()
```

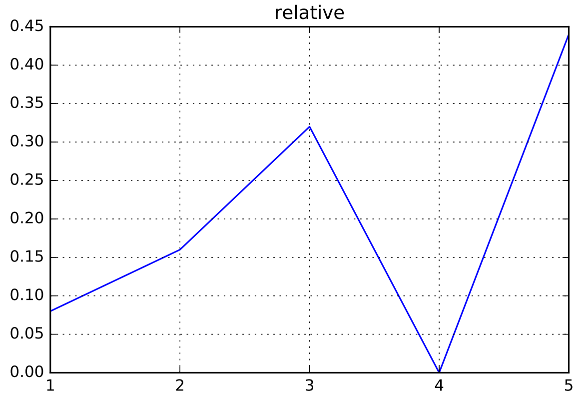

Fig. 4. problem 4(a)

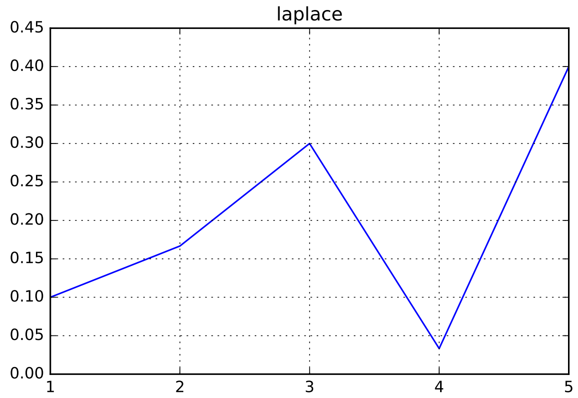

Fig. 5. problem 4(b)

From figures, we can see that increasing the number of pseudocounts makes the connected line softer. In general, it decrease the standard deviation of the

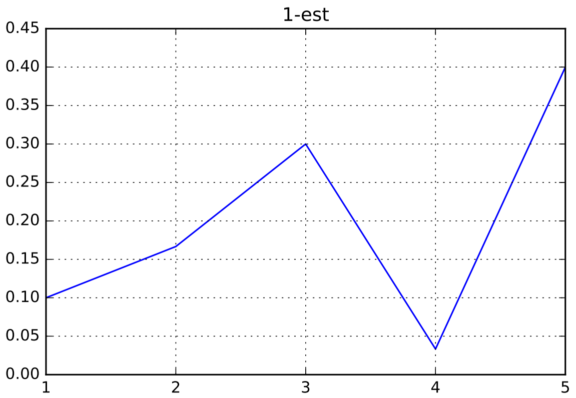

Fig. 6. problem 4(c)

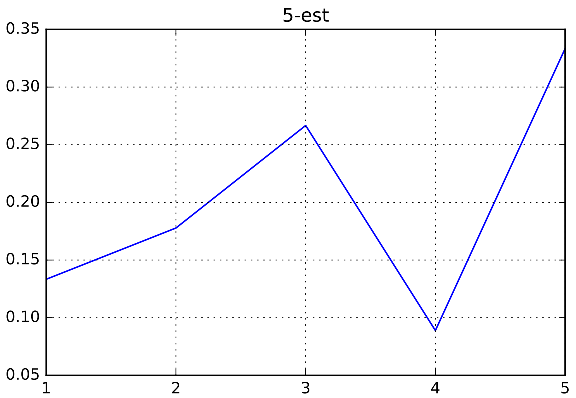

Fig. 7. problem 4(d)

pdf, that is make lower probability higher, higher probability lower. It also shows that with small dataset like this one, higher m maybe better an more general.

V. PROBLEM 5

We have Cartesian product of all features

$$4 \times 2 \times 3 \times 4 \tag{4}$$

(a)

$$2^{4*2*3*4} = 2^{96} \tag{5}$$

(b)

$$5 \times 3 \times 4 \times 5 = 300 \tag{6}$$

(c)

$$(4^2 - 1) * (2^2 - 1) * (3^2 - 1) * (4^2 - 1) = 5400 \tag{7}$$

VI. PROBLEM 6