Information Theory (UAI/500)

Lecture Notebook

Lecturer: Ivo Bukovsky

Fundamentals for Information Theory (probability, conditional probability)

- probability, independent and dependent events, joint probability conditional probability, Bayes rule, complete probability
- probability of independent and dependent events, joint probability
- conditional probability, Bayes rule, complete probability

Validation of Neural Networks (as Binary Classifiers or Detectors)

- Confusion Matrix, Sensitivity (Recall), Specificity
- n-fold cross validation
- ROC characteristics

Some "Refreshment"

- **discrete random variable:** probability mass function (pmf)
- **continuous random variable:** probability density function (pdf)

Mean Value, Variance, Standard Deviation, Modus, Median...

a) via probability, i.e., probability mass function (pmf), define:
b) via probability density function define:
***Mean Value, Variance, Standard Deviation, Modus, Median...*:**

1) Mean:

a) for discrete random variable x via pmf $p(x),\ \overline{x} = \sum_{\forall x} p(x)\cdot x$; however, practically as **sample mean** $\overline{x} = \frac{1}{N}\sum_{k=1}^{N} x_k$, where k is sample index, and N is the number of samples (data length).

b) for continous random variable x via probability density function (pdf) $f(x),\ \overline{x} = \int_{x=-\infty}^{+\infty} p(x)\cdot x\cdot dx$

2) Variance:

a) via $p(x),\ \sigma_x^2 = \sum_{\forall x} p(x)\cdot(x-\overline{x})^2 = \sum_{\forall x} p(x)\cdot x^2 - \overline{x}^2$; however, practically as:

$\sigma_x^2 = \frac{1}{N}\sum_{k=1}^{N}\left(x_k - \overline{x}\right)^2$ that estimates the variance, however, it is biased for low number of samples, i.e. N=low , so we can use **sample variance** as follows:

$\sigma_x^2 = \frac{1}{N-1}\sum_{k=1}^{N}\left(x_k - \overline{x}\right)^2$ that estimates the unbiased variance.

b) via $f(x),\ \sigma_x^2 = \int_{x=-\infty}^{+\infty} p(x)\cdot(x-\overline{x})^2 dx = \sigma_x^2 = \int_{x=-\infty}^{+\infty} p(x)\cdot x^2\cdot dx - \overline{x}^2$

3) Standard Deviation:

$\sigma_x = \sqrt{\sigma_x^2}$

4) Modus

(most frequent value) of x, i.e. - a) via pmf $p(x = x_{modus}) \overset{!}{=} max$

- b) via pdf $\frac{\partial F(x = x_{modus})}{\partial x} = max$, i.e., $f(x = x_{modus}) \overset{!}{=} max$

5) Median

(half of all data samples is bellow median, and the half is above, for even length of set, it is the mean of the two values in the middle of all smaples)

a) via pmf $F(x_{median}) = \sum_{x=x_{min}}^{x_{median}} p(x) \overset{!}{=} \frac{1}{2}$

b) via pdf $F(x_{median}) = \int_{x=x_{min}}^{x_{median}} f(x) \overset{!}{=} \frac{1}{2}$

Modus and Median Example

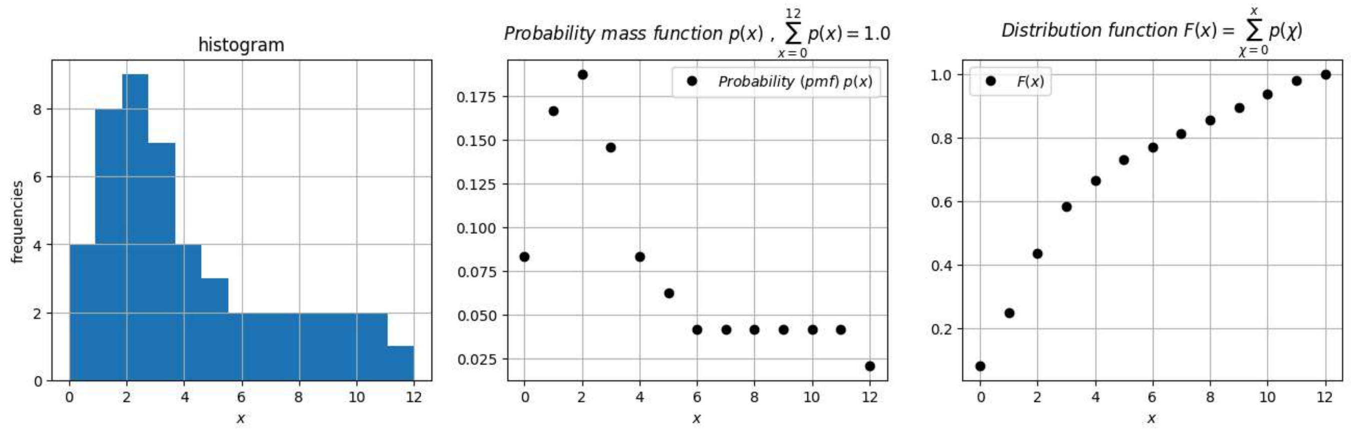
Let's have discrete random variable x that can be integer value from 0 to 12. The experiment generated its values as follows:

data set = {6,6,2,10,1,1,1,0,0,2,4,3,3,3,11,3,3,3,2,4,4,4,0,1,3,1,1,2,10,2,2,7,7,8,8,9,9,1,0,2,11,2,2,5,5,5,1,12}

To do: **draw probability, i.e., probability mass function $p(x)$, distribution function $F(x)$, calculate mean, variance, modus and median.**

(The code bellow is to show what happends, of course you can call some function for that directly)

```
%matplotlib inline
from numpy import *   # yeah, should be import numpy as np,.. I know
from matplotlib.pyplot import *

data_samples=array((6,6,2,10,1,1,1,0,0,2,4,3,3,3,11,3,3,3,2,4,4,4,0,1,3,1,1,2,10,2,2,7,7,8,8,9,9,1,0,2,11,2,2,5,5,5,1,12))

figure(figsize=(16,4))
subplot(1,3,1)
title("histogram");grid();xlabel('$x$');ylabel("frequencies")

bins=13
fb=hist(data_samples,bins)  # histogram <=> frequencies and bins

frequencies=fb[0]           # frequencies

px=frequencies/sum(frequencies)  # px=p(x)...probability, i.e. probability mass function (pmf)), x ... a dicrete random variable

x=[0,1,2,3,4,5,6,7,8,9,10,11,12]  # function values, bins for histogram can be here the same as the values of x
Fx = [sum(px[:k+1]) for k in x]  # Distribution function (for discrete x it is the cumulative probability)

subplot(132)
plot(x,px,'ok',label="$Probability \ (pmf) \ p(x)$");grid();legend();xlabel('$x$'),title("$Probability \ mass \ function \ p(x) \ ,
subplot(133)
plot(x,Fx,'ok',label="$F(x)$");grid();legend();xlabel('$x$'),title("$Distribution \ function \ F(x)=\sum_{\chi=0}^x p(\chi)$")
show()
```



```
# Mean
dataset=data_samples
print("mean(x)=", mean(dataset))
print("or")
meanx=sum([px[k]*k for k in x])
print("meanx=", meanx)
```

```
mean(x)= 4.0
or
meanx= 4.0
```

```
# Sample Variance
print("var(x)=", var(dataset))
print("or")
varx=sum([px[k]*(k-meanx)**2 for k in x])
print("var(x)=", varx)
print("or")
N=len(dataset)
samplevarx=1/(N)*sum([(k-meanx)**2 for k in dataset])
print("biased variance of x=", samplevarx)
unbsamplevarx=1/(N-1)*sum([(k-meanx)**2 for k in dataset])
print("unbiased, i.e., sample variance of x=", unbsamplevarx)
```

```
var(x)= 10.916666666666666
or
var(x)= 10.916666666666666
or
biased variance of x= 10.916666666666666
unbiased, i.e., sample variance of x= 11.148936170212766
```

```
# Modus
x=array(x)
x_modus=x[px==max(px)]
print("x_modus=",x_modus)

#Median
print("From the F(x) plot above we can see that")
Fx=array(Fx)
print("F(x=2)=",Fx[x==2])
print("F(x=3)=",Fx[x==3])
print("So the median follows as the weighted average:")
x_median=(2*px[x==2]+3*px[x==3])/(px[x==2]+px[x==3])
print(x_median)
```

```
x_modus= [2]
From the F(x) plot above we can see that
F(x=2)= [0.4375]
F(x=3)= [0.58333333]
So the median follows as the weighted average:
[2.4375]
```

Probability

$$P(A,B) \text{ vs. } P(A|B), P(B|A)$$

Tossings Two Coins

A...coin A turns head , $\overline{A}$ ... coin A turns tail (complement to event A)
B ... coin B turns head, $\overline{B}$ ... coin B turns tail (complement to event B)

1. What is probability that when tossing both (fair) coins, we obtain:
   - two heads?
   - two tails?
   - coin A turns head and coin B turns tale?
   - head and tale(regardless which coin)?
   (two tails, ?
2. When coin B shows head, what is notation for probability that coin A shows tail?
3. When coin A shows tail, what is notation for probability that coin B shows head?
4. When coin A shows , tail, what is notation for probability that coin B shows tail?
5. Why conditional probability does not make much sense in this problem?

Throwing Two Dices

A...dice A shows "6" , $\overline{A}$ ... dice A does not show "6" but "1" or "2" or "3" or "4" or "5" (complement to event A)
B...dice B shows "6", $\overline{B}$ ... dice B does not show "6" but "1" or "2" or "3" or "4" or "5" (complement to event B)

1. What is probability that when throwing both (fair) dices, we obtain:
   - two "6"?
   - no "6"?
   - dice A shows "6" and coin B does not show "6"?
   - "6" only once (regardless which coin)?
2. When dice B showed "6", what is notation for probability that dice A shows "6"?
3. When dice A showed "6", what is notation for probability that dice B shows "6"?
4. When dice A does not show "6", what is notation for probability that dice B shows tail?

5. Why conditional probability does not make much sense in this problem?

The Bayes Rule (and Complete Probability)

A Single Flying Drone (example)

1. A randomly flying drone is lost:
   A ... the drone crashed into zone A, $\overline{A}$ ...the drone did not crash into zone A
   B ... the drone crashed into zone B, $\overline{B}$ ...the drone did not crash into zone B
   (C ... the drone crashed into Zone $C, \overline{C}$ ...)
   a) there are sharp borders between zones (zones do not overlap)
   b) the zones overlap **(Conditional Probability and Bayes Rule)**

Connotate the situations to the following sketches and to probabilities as $P(A \cap B), P(A \cap B \cap C), P(A \cup B), P(A \cup B \cup C, P(A|B), P(B|A)$
notice notation: $P(A \cap B) = P(A, B) = P(AB)$.

image-2.png image-3.png image-4.png

Example L-4

A SW company received 1 000 calls from customers who use operating system (OS) A and 2 000 calls from customers who run OS B (per year).
It was observed that only 100 OS A customers and 120 OS B customers had trully serious problem.
Asuming that the problems of customers are independent events, what is the probability that the next calling customer:

1. has a serious problem?
2. runs OS A?
3. runs OS B?
4. runs OS A and has a serious problem? **[0.02 Points]**
5. runs OS B and has a serious problem? **[0.02 Points]**
6. has a serious problem, while we allready know the customer runs OS A? **[0.1 Points]**
7. has a serious problem, while we allready know the customer runs OS B? **[0.1 Points]**
8. runs OS A, while we allready know the customer has a serious problem?**[0.1 Points]**
9. runs OS B, while we allready know the customer has a serious problem? **[0.1 Points]**
10. does not have a serious problem, while we allready know the customer runs OS A? **[0.1 Points]**
11. does not have a problem, while we allready know the customer runs OS B? **[0.1 Points]**
12. runs OS A, while we allready know the customer does not have a serious problem?**[0.1 Points]**
13. runs OS B, while we allready know the customer does not have a serious problem?**[0.1 Points]**

```python
# Given data
total_A = 1000
total_B = 2000
serious_A = 100
serious_B = 120

total_customers = total_A + total_B
total_serious_prob = serious_A + serious_B
total_not_serious = total_customers - total_serious_prob

# what is the probability that the next calling customer:
# 1) has a serious problem?
P_serious_problem = total_serious_prob / total_customers
print("P(serious problem) =", P_serious_problem)

# 2) runs OS A?
P_A = total_A / total_customers
print("P(OS A) =", P_A)

# 3) runs OS B?
P_B = total_B / total_customers
print("P(OS B) =", P_B)

# 4) runs OS A and has a serious problem?
P_A_and_serious = serious_A / total_customers
print("P(OS A and serious) =", P_A_and_serious)

# 5) runs OS B and has a serious problem?
```

```python
P_B_and_serious = serious_B / total_customers
print("P(OS B and serious) =", P_B_and_serious)

# 6) has a serious problem, while we allready know the customer runs OS A?
P_serious_given_A = serious_A / total_A
print("P(serious | OS A) =", P_serious_given_A)

# 7) has a serious problem, while we allready know the customer runs OS B?
P_serious_given_B = serious_B / total_B
print("P(serious | OS B) =", P_serious_given_B)

# 8) runs OS A, while we allready know the customer has a serious problem?
P_A_given_serious = serious_A / total_serious_prob
print("P(OS A | serious) =", P_A_given_serious)

# 9) runs OS B, while we allready know the customer has a serious problem?
P_B_given_serious = serious_B / total_serious_prob
print("P(OS B | serious) =", P_B_given_serious)

# 10) does not have a serious problem, while we allready know the customer runs OS A?
P_not_serious_given_A = (total_A - serious_A) / total_A
print("P(not serious | OS A) =", P_not_serious_given_A)

# 11) does not have a problem, while we allready know the customer runs OS B?
P_not_serious_given_B = (total_B - serious_B) / total_B
print("P(not serious | OS B) =", P_not_serious_given_B)

# 12) runs OS A, while we allready know the customer does not have a serious problem?
P_A_given_not_serious = (total_A - serious_A) / total_not_serious
print("P(OS A | not serious) =", P_A_given_not_serious)

# 13) runs OS B, while we allready know the customer does not have a serious problem?
P_B_given_not_serious = (total_B - serious_B) / total_not_serious
print("P(OS B | not serious) =", P_B_given_not_serious)
```

```
P(serious problem) = 0.07333333333333333
P(OS A) = 0.3333333333333333
P(OS B) = 0.6666666666666666
P(OS A and serious) = 0.03333333333333333
P(OS B and serious) = 0.04
P(serious | OS A) = 0.1
P(serious | OS B) = 0.06
P(OS A | serious) = 0.45454545454545453
P(OS B | serious) = 0.5454545454545454
P(not serious | OS A) = 0.9
P(not serious | OS B) = 0.94
P(OS A | not serious) = 0.3237410071942446
P(OS B | not serious) = 0.6762589928057554
```

Example L-5

Let's have two sources A and B, that send their messages towards receiver R via separate very noisy communication channels Ch_A and Ch_B .

Source A sends 10 000 messages, and source B sends 20 000 messages.

Fig.2: Noisy communication channel example.

We know from the past, that channel Ch_A disrupts 5% of all messages from source A and channel Ch_B disrupts 1% of all messages from B.

Questions:

Assumption: The disruptions of messages are independent (also in time), i.e., the probability that the next message is disrupted is the same as that one of the previous messages.

1. The receiver R receives a message, what is probability that:
   - the message is from A?
   - the message is from B?
   - the message is ok?**[0.1 Points]**
   - the message is disrupted? **[0.1 Points]**
2. Generally, what is probability that:
   - a message from A is disrupted?**[0.1 Points]**
   - a message from A is ok?**[0.1 Points]**
   - a message from B is disrupted?**[0.1 Points]**
   - a message from B is ok?**[0.1 Points]**
   - a disrupted message is from A?**[0.1 Points]**
   - an ok message is from A?**[0.1 Points]**

- a disrupted message is from B?**[0.1 Points]**
- an ok message is from B? **[0.1 Points]**

3. - The receiver R receives 100 new messages, how many messages is most probably disrupted?**[0.2 Points]**
   - What is the probability that 0 or 1 or 2 or 3... or 100 messages are disrupted?**[0.3 Points]**
   - What is the probability that 0 or max 1 or max 2 or max 3... or max 100 messages are disrupted?**[0.4 Points]**

```python
import numpy as np
from scipy.stats import binom

messages_A = 10000
messages_B = 20000
disruption_A = 0.05  # 5% disruption for messages from A
disruption_B = 0.01  # 1% disruption for messages from B

total_messages = messages_A + messages_B

# 1) The receiver receives a message, what is probability that
# a) the message is from A
P_A = messages_A / total_messages
print("P(A) =", P_A)
# b) the message is from B
P_B = messages_B / total_messages
print("P(B) =", P_B)
# c) the message is ok?
P_OK_A = 1 - disruption_A  # Probability that a message from A is OK
P_OK_B = 1 - disruption_B  # Probability that a message from B is OK

# Using the law of total probability to calculate P(OK)
P_OK = P_OK_A * P_A + P_OK_B * P_B
print("P(OK) =", P_OK)

# d) the message is disrupted?
P_disrupted = 1 - P_OK
print("P(disrupted) =", P_disrupted)


# 2) Generally, what is probability that:
# a) a message from A is disrupted?
P_disrupted_given_A = disruption_A
print("P(disrupted | A) =", P_disrupted_given_A)

#b) a message from A is ok?
P_okay_A = 1 - P_disrupted_given_A
print("P(OK | A) =", P_okay_A)

# c) a message from B is disrupted?
P_disrupted_given_B = disruption_B
print("P(disrupted | B) =", P_disrupted_given_B)

#d) a message from B is ok?
P_okay_B = 1 - P_disrupted_given_B
print("P(OK | B) =", P_okay_B)

# e) a disrupted message is from A?
# Total probability that a message is disrupted
P_disrupted = (P_disrupted_given_A * P_A) + (P_disrupted_given_B * P_B)
P_A_given_disrupted = (P_disrupted_given_A * P_A) / P_disrupted
print("P(A | disrupted) =", P_A_given_disrupted)

# f) an ok message is from A?
# Total probability that a message is OK
P_OK = 1 - P_disrupted
P_A_given_OK = (P_okay_A * P_A) / P_OK
print("P(A | OK) =", P_A_given_OK)

# g) a disrupted message is from B?
P_B_given_disrupted = (P_disrupted_given_B * P_B) / P_disrupted
print("P(B | disrupted) =", P_B_given_disrupted)

# h) an ok message is from B?
P_B_given_OK = (P_okay_B * P_B)/P_OK
print("P(B | OK) =", P_B_given_OK)


# 3) a)The receiver  R  receives 100 new messages, how many messages is most probably disrupted?
P_disrupted_100 = P_disrupted * 100
print("Probability of disrupted messages =", np.floor(P_disrupted_100))

```

```
# b) What is the probability that 0 or 1 or 2 or 3... or 100 messages are disrupted?
# The probability of exactly k disrupted messages is given by binomial distribution
p_k = []
for k in range(101):
  prob = binom.pmf(k,100,P_disrupted)
  p_k.append(prob)
print("Probability of exactly k disrupted messages (first 10 values):", np.round(p_k[:10],5))

# c) What is the probability that 0 or max 1 or max 2 or max 3... or max 100 messages are disrupted?
# We need to calculate the cumulative probability that 0, 1, 2, ..., or 100 messages are disrupted.
# The cumulative probability is the sum of binomial probabilities up to a certain number k

cumulative_prob = []
for k in range(101):
  prob = binom.cdf(k,100,P_disrupted)
  cumulative_prob.append(prob)
print("Cumulative probability (first 10 values):", np.round(cumulative_prob[:10],5))
```

```
P(A) = 0.3333333333333333
P(B) = 0.6666666666666666
P(OK) = 0.9766666666666666
P(disrupted) = 0.023333333333333428
P(disrupted | A) = 0.05
P(OK | A) = 0.95
P(disrupted | B) = 0.01
P(OK | B) = 0.99
P(A | disrupted) = 0.7142857142857143
P(A | OK) = 0.3242320819112628
P(B | disrupted) = 0.2857142857142857
P(B | OK) = 0.6757679180887372
Probability of disrupted messages = 2.0
Probability of exactly k disrupted messages (first 10 values): [0.09433 0.22535 0.2665  0.20799 0.1205  0.05527 0.02091 0.00671 0.0
 0.00045]
Cumulative probability (first 10 values): [0.09433 0.31968 0.58619 0.79417 0.91467 0.96994 0.99085 0.99756 0.99942
 0.99988]
```

Validation of Neural Networks (as Binary Classifiers or Detectors)

- Confusion Matrix, Sensitivity (Recall), Specificity
- n-fold cross validation (briefly)
- ROC curves (briefly)

A Neural Network as a Binary Detector (example)

A neural network is trained to detect if a car is in the image or not. There is 10 000 images, 4 000 images contain a car, 6 000 images show no car. The method failed to find cars in 200 images with a car, and also it incorrectly detected cars in 100 images without any car.

Let's denote notation for discrete, here also binary, events as follows:

REALITY: A ... a car is really in the image, $\overline{A}$ ...the image is without a car
METHOD: B ... the neural network is correct, $\overline{B}$ ...the neural network is wrong

1. The graphical notion of probabilities and Bayes Rule connotations
2. The Confusion Matrix
3. Sensitivity(Recall) and Specificity of the detector

image-2.png

Figure 1: This Confusion Matrix is adopted from [e] as it is clear and nice summary(occasionaly, for the exam, you do not have to remember the names and formulas of all parameters (focus mainly on TP, FN, TN,FP, FPR, TNR, TPR, FNR) , to understand what is it for matters!, see also [a] [b] to be "correct"

A Neural Network as a Binary Detector (extension with the detection bias)

In practice, the classification or detection methods depends on some customable bias.

E.g.:

- if the bias value is exceeded, the result of the detection method is Positive,
- if the bias value is not exceeded, the result of the detection method is Negative.

Let's denote $\beta \in \langle \beta_{min}, \beta_{max} \rangle$ denote a detection bias. Then a point in 2-D that is drawn as $[x, y] = [FPR(\beta), TPR(\beta)]$ represents a single point of the **Receiver Operating Characterstics (ROC)** [e].

Thus, ROC is drawn as the $[x, y]\ \forall \beta$.

See, e.g., [e] , for an image diagnosis example you may see eg. [h])

image-4.png

Figure 2: The principle of the Receiver Operating Characteristic curves (source [e] has it done quite nicely, see Fig. 1 for meaning of axes).

Example L-6

Receiver R receives many messages. The receiver applies a detection method to recognize whether the received messages are OK or disrupted (NOK).

However, the detection method is imperfect. The performance of the detection method was validated on 1 000 messages for a particular bias setup, where we allready know that 300 messages were received with disruption.

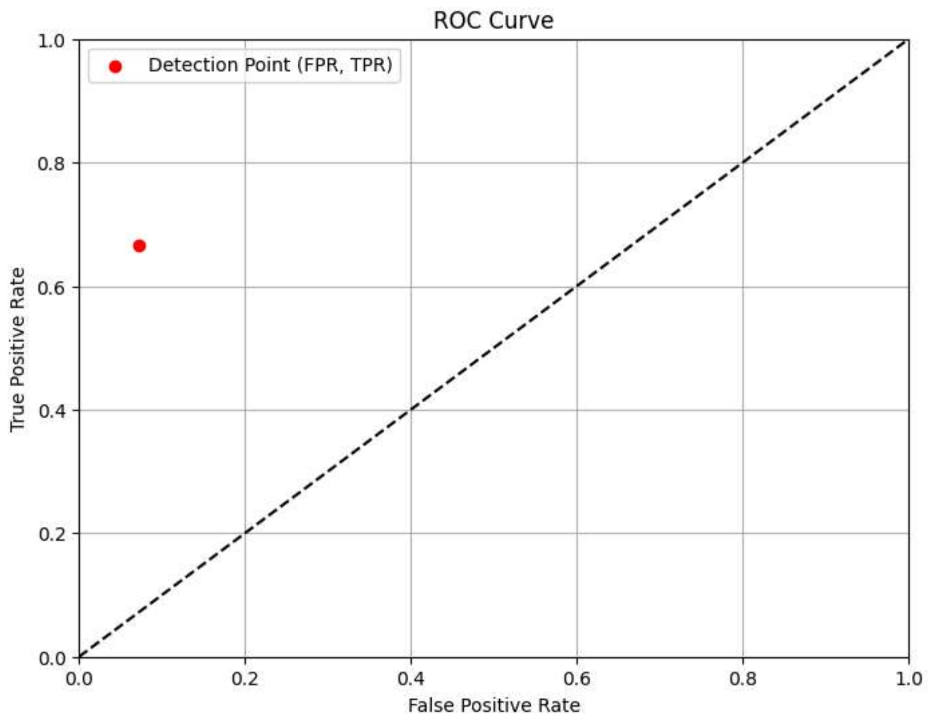
It was found that the detection method:

- detected correctly only 200 disrupted messages,
- made mistake with 50 correctly received messages, i.e., the method detected 50 OK messages as NOK (Not OK) messages.

1. Sketch confusion matrix of the detection method **[0.2 Points]**
2. What is the $\mathbf{sensitivity}$ of the detection method?**[0.4 Points]**
3. What is the $\mathbf{specificity}$ of the detection method?**[0.4 Points]**
4. Draw the point of **ROC** for the given parameters (as for the single setup of detection bias) **[0.4 Points]**

```python
import matplotlib.pyplot as plt
import pandas as pd
#1) Sketch confusion matrix of the detection method?
total_messages = 1000
disrupted_messages = 300
ok_messages = total_messages - disrupted_messages  # OK messages

true_positive = 200 # Correctly detected disrupted messages
false_positive = 50 # OK messages detected as NOK

false_negative = disrupted_messages - true_positive # NOK incorrectly detected as OK
true_negative = ok_messages - false_positive  #OK messages correctly detected

confusion_matrix = pd.DataFrame({
    'Detected OK': [true_negative, false_positive],
    'Detected NOK': [false_negative, true_positive]
}, index=['Actual OK', 'Actual NOK'])

print("Confusion Matrix:")
print(confusion_matrix)

# 2) What is the sensitivity of the detection method?

sensitivity = true_positive / (true_positive + false_negative)
print(f"\nSensitivity of the detection method: {sensitivity:.4f}")

# 3) what is the specificity of the detection method?

specificity = true_negative / (true_negative + false_positive)
print(f"\nSpecificity of the detection method: {specificity:.4f}")

# 4) Draw the point of ROC for the given parameters?

TPR = sensitivity  # True Positive Rate (Sensitivity)
FPR = false_positive / (false_positive + true_negative)  # False Positive Rate

plt.figure(figsize=(8,6))
plt.plot([0,1],[0,1],'k--')
plt.scatter(FPR,TPR,color='red', label='Detection Point (FPR, TPR)')
plt.xlim(0,1)
plt.ylim(0,1)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend()
plt.grid()
plt.show()
```

```
Confusion Matrix:
            Detected OK  Detected NOK
Actual OK           650           100
Actual NOK           50           200

Sensitivity of the detection method: 0.6667

Specificity of the detection method: 0.9286
```



Points and Assigments (week 2):

If you wish to submit your solutions to collect your optional points, solve the problems in this notebook directly and upload to Moodle within 2 weeks.

In this notebook, you may collect maximum of 3 points (though the total sum of points available in this notebook is more, so you have a lot of chance for 3 ;).

References

[a]
“Wikipedia:Academic use,” Wikipedia. Feb. 17, 2021. Accessed: Oct. 13, 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Wikipedia:Academic_use&oldid=1007392296
[b]
“Wikipedia:Text of Creative Commons Attribution-ShareAlike 3.0 Unported License,” Wikipedia. Sep. 23, 2021. Accessed: Oct. 13, 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Wikipedia:Text_of_Creative_Commons_Attribution-ShareAlike_3.0_Unported_License&oldid=1045983641
[c]
“Confusion matrix,” Wikipedia. Jul. 04, 2021. Accessed: Oct. 13, 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Confusion_matrix&oldid=1031861694
[d]
“Evaluation of binary classifiers,” Wikipedia. May 09, 2021. Accessed: Oct. 13, 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Evaluation_of_binary_classifiers&oldid=1022265108
[e]
“Receiver operating characteristic,” Wikipedia. Oct. 09, 2021. Accessed: Oct. 13, 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Receiver_operating_characteristic&oldid=1049030439
[f]
J. Brownlee, “A Gentle Introduction to Bayes Theorem for Machine Learning,” Machine Learning Mastery, Oct. 03, 2019. https://machinelearningmastery.com/bayes-theorem-for-machine-learning/ (accessed Oct. 13, 2021).
[g]
J. L. Puga, M. Krzywinski, and N. Altman, “Bayes’ theorem,” Nature Methods, vol. 12, no. 4, pp. 277–278, Apr. 2015, doi: 10.1038/nmeth.3335.
[h]
N. Homma et al., “A Deep Learning Aided Drowning Diagnosis for Forensic Investigations using Post-Mortem Lung CT Images,” in 2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC), Montreal, QC, Canada, Jul. 2020, pp. 1262–1265. doi: 10.1109/EMBC44109.2020.9175731.

Basic Reading:

[1] MACKAY, David J. C. Information theory, inference, and learning algorithms. Cambridge: Cambridge University Press, 2003. ISBN 978-0-521-64298-9..

Recommended Reading:

[2] COVER, T. M. and Joy A. THOMAS. Elements of information theory. 2nd ed. Hoboken: Wiley-Interscience, c2006. ISBN 978-0-471-24195-9..
[3] HOST, S. Information and Communication Theory. Hoboken, NJ: Wiley-IEEE Press, 2019. ISBN 978-1119433781..
[4] EL-GAMAL, A. and YOUNG-HAN, K. Network information theory. Primera. Cambridge: Cambridge University Press, 2011. ISBN 978-1-107-00873-1..

© Ivo Bukovsky, Dpt. of Comp. Sci., USB in CB, 2021