Information Theory (UAI/500)

No description has been provided for this image

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 $p(x)$, $\overline{x} = \sum\limits_{\forall x} p(x) \cdot x$; however, practically as **sample mean** $\overline{x} = \frac{1}{N} \sum\limits_{k=1}^{N} x_k$, 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) $f(x)$, $\overline{x} = \int\limits_{x=-\infty}^{+\infty} p(x) \cdot x \cdot dx$

2) Variance:

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

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

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

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 $\dfrac{\partial F(x = x_{modus})}{\partial x} = max$, 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 $F(x_{median}) = \sum\limits_{x=x_{min}}^{x_{median}} p(x) \overset{!}{=} \dfrac{1}{2}$
b) via pdf $F(x_{median}) = \int\limits_{x=x_{min}}^{x_{median}} f(x) \overset{!}{=} \dfrac{1}{2}$

Modus and Median Example

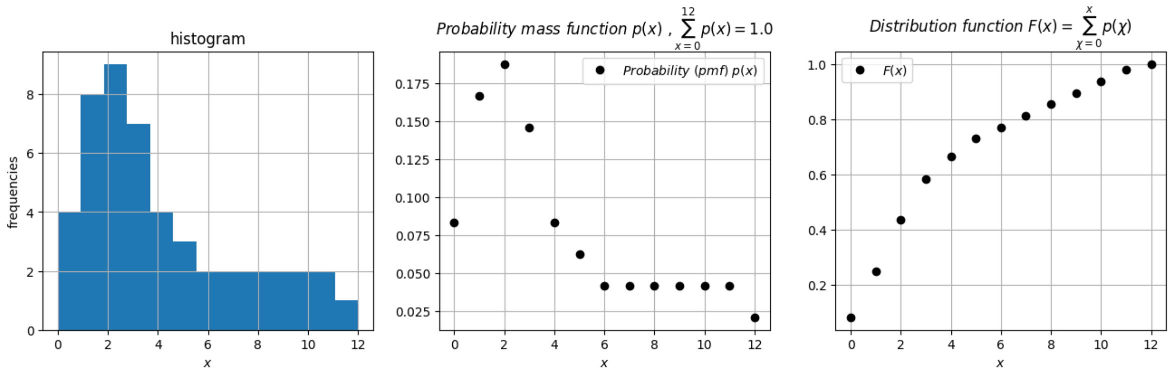
Let's have discrete random variable x that can be integer value from 0 to 12. The experiment generated its values as follows:

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)

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

data_samples=array((6,6,2,10,1,1,1,0,0,2,4,3,3,3,11,3,3,3,2,4,4,4,0,1,3,1,1,2,1

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

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

subplot(132)
plot(x,px,'ok',label="$Probability \ (pmf) \ p(x)$");grid();legend();xlabel('$x$
subplot(133)
plot(x,Fx,'ok',label="$F(x)$");grid();legend();xlabel('$x$'),title("$Distributio
show()
```



In [ ]:
```
# 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
```

In [ ]:
```
# 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
```

In [ ]:
```
# 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?

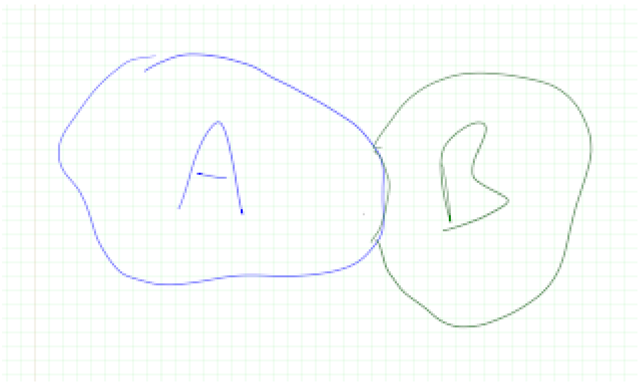
The Bayes Rule (and Complete Probability)

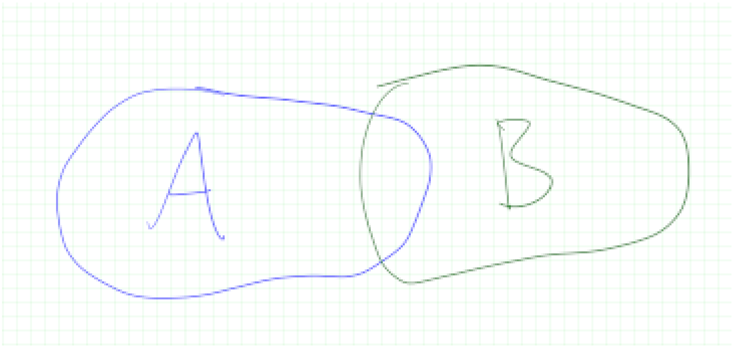
A Single Flying Drone (example)

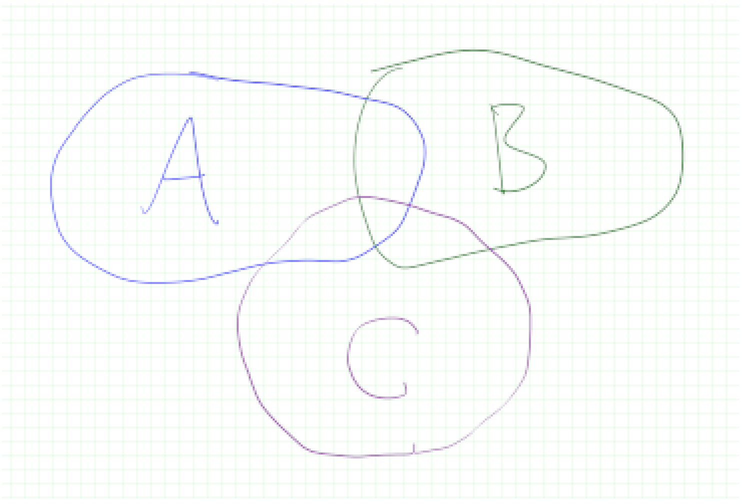
1. A randomly flying drone is lost:
   A ... the drone crashed into zone A, $\overline{A}$ ...the drone did not crash into zone A
   B ... the drone crashed into zone B, $\overline{B}$ ...the drone did not crash into zone B
   (C ... the drone crashed into Zone C, $\overline{C}$ ...)
   a) there are 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: $P(A \cap B) = P(A, B) = P(AB)$.







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]**

In [ ]:

```
# runs OS A and has a serious problem?
# runs OS B and has a serious problem?
# has a serious problem, while we allready know the customer runs OS A?
# has a serious problem, while we allready know the customer runs OS B?
# runs OS A, while we allready know the customer has a serious problem?
# runs OS B, while we allready know the customer has a serious problem?
# does not have a serious problem, while we allready know the customer runs OS A
# does not have a problem, while we allready know the customer runs OS B?
# runs OS A, while we allready know the customer does not have a serious problem
# runs OS B, while we allready know the customer does not have a serious problem

# Given data
total_calls_A = 1000
total_calls_B = 2000
calls_serious_A = 100
calls_serious_B = 120

# Probability of having a serious problem (P(S))
P_S = (calls_serious_A + calls_serious_B) / (total_calls_A + total_calls_B)
print("Probability of having a serious problem (P(S)): ", P_S)

# Probability of running OS A (P(A))
P_A = total_calls_A / (total_calls_A + total_calls_B)
print("Probability of running OS A (P(A)): ", P_A)

# Probability of running OS B (P(B))
P_B = total_calls_B / (total_calls_A + total_calls_B)
print("Probability of running OS B (P(B)): ", P_B)

# Probability of running OS A and having a serious problem (P(A and S))
P_A_and_S = (P_A) * (calls_serious_A / total_calls_A)
print("Probability of running OS A and having a serious problem (P(A and S)): ",

# Probability of running OS B and having a serious problem (P(B and S))
P_B_and_S = (P_B) * (calls_serious_B / total_calls_B)
print("Probability of running OS B and having a serious problem (P(B and S)): ",

# Probability of having a serious problem given the customer runs OS A (P(S|A))
P_S_given_A = calls_serious_A / total_calls_A
print("Probability of having a serious problem given the customer runs OS A (P(S

# Probability of having a serious problem given the customer runs OS B (P(S|B))
P_S_given_B = calls_serious_B / total_calls_B
print("Probability of having a serious problem given the customer runs OS B (P(S

# Probability of running OS A given the customer has a serious problem (P(A|S))
```

```
P_A_given_S = (P_A) * (P_S_given_A) / P_S
print("Probability of running OS A given the customer has a serious problem (P(A

# Probability of running OS B given the customer has a serious problem (P(B|S))
P_B_given_S = (P_B) * (P_S_given_B) / P_S
print("Probability of running OS B given the customer has a serious problem (P(B

# Probability of not having a serious problem given the customer runs OS A (P(no
P_not_S_given_A = 1 - P_S_given_A
print("Probability of not having a serious problem given the customer runs OS A

# Probability of not having a serious problem given the customer runs OS B (P(no
P_not_S_given_B = 1 - P_S_given_B
print("Probability of not having a serious problem given the customer runs OS B

#runs OS A, while we allready know the customer does not have a serious problem
P_not_S = 1 - P_S
P_A_given_not_S = (P_not_S_given_A * P_A)/ P_not_S
print("Probability of running OS A, while we allready know the customer does not

#runs OS B, while we allready know the customer does not have a serious problem
P_B_given_not_S = (P_not_S_given_B * P_B)/ P_not_S
print("Probability of running OS B, while we allready know the customer does not
```

```
Probability of having a serious problem (P(S)):  0.07333333333333333
Probability of running OS A (P(A)):  0.3333333333333333
Probability of running OS B (P(B)):  0.6666666666666666
Probability of running OS A and having a serious problem (P(A and S)):  0.0333333
3333333333
Probability of running OS B and having a serious problem (P(B and S)):  0.0399999
99999999994
Probability of having a serious problem given the customer runs OS A (P(S|A)):
0.1
Probability of having a serious problem given the customer runs OS B (P(S|B)):
0.06
Probability of running OS A given the customer has a serious problem (P(A|S)):
0.45454545454545453
Probability of running OS B given the customer has a serious problem (P(B|S)):
0.5454545454545454
Probability of not having a serious problem given the customer runs OS A (P(not S
|A)):  0.9
Probability of not having a serious problem given the customer runs OS B (P(not S
|B)):  0.94
Probability of running OS A, while we allready know the customer does not have a
serious problem 0.3237410071942446
Probability of running OS B, while we allready know the customer does not have a
serious problem 0.6762589928057553
```

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.

No description has been provided for this image

Fig.2: Noisy communication channel example.

In [ ]:
```
# 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]
# 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]
# - The receiver  R  receives 100 new messages, how many messages is most probab
# - What is the probability that 0 or 1 or 2 or 3... or 100 messages are disrupt
# - What is the probability that 0 or max 1 or max 2 or max 3... or max 100 mess

# Given data
total_messages_A = 10000
total_messages_B = 20000
disruption_rate_A = 0.05  # 5% disruption rate for source A
disruption_rate_B = 0.01  # 1% disruption rate for source B
```

```python
# Calculate probabilities
P_A = total_messages_A / (total_messages_A + total_messages_B)
P_B = total_messages_B / (total_messages_A + total_messages_B)
P_Ok = 1 - (disruption_rate_A + disruption_rate_B) # Probability of a message be
P_Disrupted = 1 - P_Ok  # Probability of a message being disrupted

# Probability that a message is from A
P_A_message = P_A

# Probability that a message is from B
P_B_message = P_B

# Probability that a message is ok
P_Ok_message = P_Ok

# Probability that a message is disrupted
P_Disrupted_message = P_Disrupted

# Probability that a message from A is disrupted
P_A_disrupted = P_A * disruption_rate_A

# Probability that a message from A is ok
P_A_ok = P_A * (1 - disruption_rate_A)

# Probability that a message from B is disrupted
P_B_disrupted = P_B * disruption_rate_B

# Probability that a message from B is ok
P_B_ok = P_B * (1 - disruption_rate_B)

# Probability that a disrupted message is from A
P_Disrupted_from_A = (P_A_disrupted) / (P_A_disrupted + P_B_disrupted)

# Probability that an ok message is from A
P_Ok_from_A = (P_A_ok) / (P_A_ok + P_B_ok)

# Calculate probabilities for receiving 100 new messages that are disrupted
from scipy.stats import binom

# Expected number of disrupted messages (mean)
expected_disrupted = P_Disrupted_message * 100

# Probability mass function for the number of disrupted messages
disrupted_probs = [binom.pmf(k, 100, P_Disrupted_message) for k in range(101)]

# Probability that 0 or 1 or 2... or 100 messages are disrupted
cumulative_probs = [sum(disrupted_probs[:i+1]) for i in range(101)]

# Probability that 0 or max 1 or max 2... or max 100 messages are disrupted
max_disrupted_probs = [cumulative_probs[0]] + [cumulative_probs[i] - cumulative_

# Print the results
print("Probability that a message is ok: ", P_Ok_message)
print("Probability that a message is disrupted: ", P_Disrupted_message)
print("Probability that a message from A is disrupted: ", P_A_disrupted)
print("Probability that a message from A is ok: ", P_A_ok)
print("Probability that a message from B is disrupted: ", P_B_disrupted)
print("Probability that a message from B is ok: ", P_B_ok)
print("Probability that a disrupted message is from A: ", P_Disrupted_from_A)
print("Probability that an ok message is from A: ", P_Ok_from_A)
print("Expected number of disrupted messages out of 100: ", expected_disrupted)
print("Probability that 0 or 1 or 2... or 100 messages are disrupted: ", cumulat
print("Probability that 0 or max 1 or max 2... or max 100 messages are disrupted
```

```
Probability that a message is from A:  0.3333333333333333
Probability that a message is from B:  0.6666666666666666
Probability that a message is ok:  1
Probability that a message is disrupted:  0
Probability that a message from A is disrupted:  0.016666666666666666
Probability that a message from A is ok:  0.31666666666666665
Probability that a message from B is disrupted:  0.006666666666666666
Probability that a message from B is ok:  0.6599999999999999
Probability that a disrupted message is from A:  0.7142857142857143
Probability that an ok message is from A:  0.3242320819112628
Expected number of disrupted messages out of 100:  0
Probability that 0 or 1 or 2... or 100 messages are disrupted:  [1.0, 1.0, 1.0,
1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
1.0, 1.0]
Probability that 0 or max 1 or max 2... or max 100 messages are disrupted:  [1.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0]
```

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)

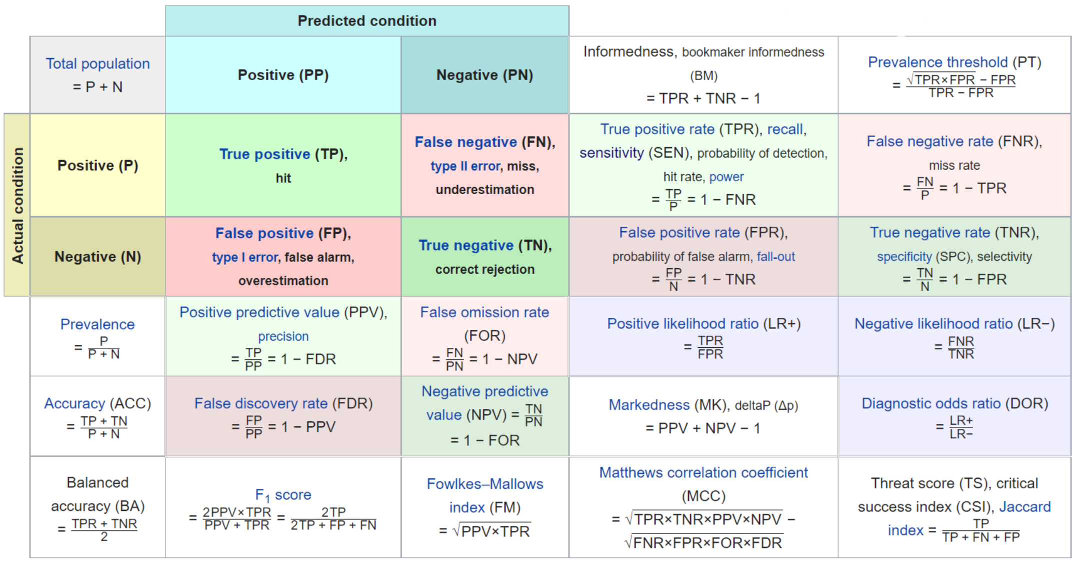
A neural network is trained to detect if a car is in the image or not. There is 10 000 images, 4 000 images contain a car, 6 000 images show no car. The method failed to find cars in 200 images with a car, and also it incorrectly detected cars in 100 images without any car.

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



		Predicted condition			
	Total population $= P + N$	**Positive (PP)**	**Negative (PN)**	Informedness, bookmaker informedness (BM) $= TPR + TNR - 1$	Prevalence threshold (PT) $= \frac{\sqrt{TPR \times FPR} - FPR}{TPR - FPR}$
Actual condition	**Positive (P)**	**True positive (TP),** **hit**	**False negative (FN),** **type II error, miss,** **underestimation**	True positive rate (TPR), recall, sensitivity (SEN), probability of detection, hit rate, power $= \frac{TP}{P} = 1 - FNR$	False negative rate (FNR), miss rate $= \frac{FN}{P} = 1 - TPR$
	Negative (N)	**False positive (FP),** **type I error, false alarm,** **overestimation**	**True negative (TN),** **correct rejection**	False positive rate (FPR), probability of false alarm, fall-out $= \frac{FP}{N} = 1 - TNR$	True negative rate (TNR), specificity (SPC), selectivity $= \frac{TN}{N} = 1 - FPR$
	Prevalence $= \frac{P}{P + N}$	Positive predictive value (PPV), precision $= \frac{TP}{PP} = 1 - FDR$	False omission rate (FOR) $= \frac{FN}{PN} = 1 - NPV$	Positive likelihood ratio (LR+) $= \frac{TPR}{FPR}$	Negative likelihood ratio (LR−) $= \frac{FNR}{TNR}$
	Accuracy (ACC) $= \frac{TP + TN}{P + N}$	False discovery rate (FDR) $= \frac{FP}{PP} = 1 - PPV$	Negative predictive value (NPV) $= \frac{TN}{PN}$ $= 1 - FOR$	Markedness (MK), deltaP (Δp) $= PPV + NPV - 1$	Diagnostic odds ratio (DOR) $= \frac{LR+}{LR-}$
	Balanced accuracy (BA) $= \frac{TPR + TNR}{2}$	F_1 score $= \frac{2PPV \times TPR}{PPV + TPR} = \frac{2TP}{2TP + FP + FN}$	Fowlkes–Mallows index (FM) $= \sqrt{PPV \times TPR}$	Matthews correlation coefficient (MCC) $= \sqrt{TPR \times TNR \times PPV \times NPV} - \sqrt{FNR \times FPR \times FOR \times FDR}$	Threat score (TS), critical success index (CSI), Jaccard index $= \frac{TP}{TP + FN + FP}$

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)

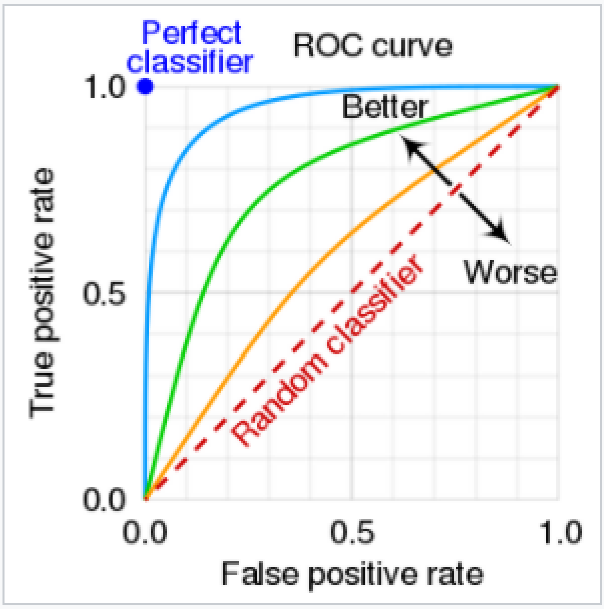
In practice, the classification or detection methods depends on some customable bias.

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])



The ROC space for a "better" and "worse" classifier.

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.

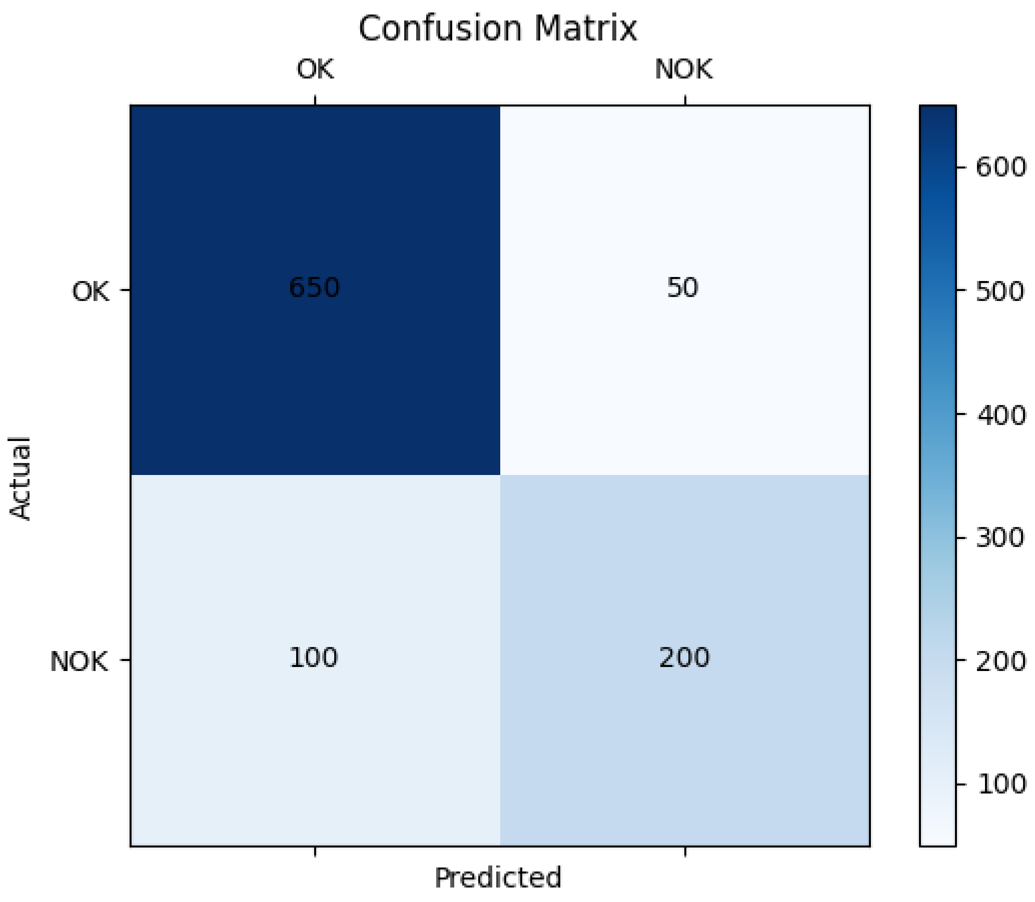
It was found that the detection method:

- 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 $\textbf{sensitivity}$ of the detection method?**[0.4 Points]**
3. What is the $\textbf{specificity}$ 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]**

In [ ]:
```
#1 Sketch confusion matrix of the detection method [0.2 Points]

#FN = detected correctly only 200 disrupted messages
#TN = made mistake with 50 correctly received messages
#FP = total messages received with disruption (300) - FN (detected correctly onl
#TP = total messages (1000)- (FN+TN+FP)
```

```
import matplotlib.pyplot as plt

# Values for the confusion matrix
TP = 650
TN = 50
FP = 100
FN = 200

# Create the confusion matrix plot
fig, ax = plt.subplots()
cax = ax.matshow([[TP, TN], [FP, FN]], cmap=plt.cm.Blues)

# Add labels and annotations
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.xticks([0, 1], ['OK', 'NOK'])
plt.yticks([0, 1], ['OK', 'NOK'])
plt.colorbar(cax)

# Display the values in the matrix
for i in range(2):
    for j in range(2):
        plt.text(j, i, str([[TP, TN], [FP, FN]][i][j]), va='center', ha='center'

plt.title('Confusion Matrix')
plt.show()
```



In [ ]:
```
#2 What is the  sensitivity  of the detection method?
TP = 650
FN = 200
```

```
# Calculate sensitivity
sensitivity = TP / (TP + FN)

# Print sensitivity
print("Sensitivity (True Positive Rate):", sensitivity)
```

```
Sensitivity (True Positive Rate): 0.7647058823529411
```

In [ ]:

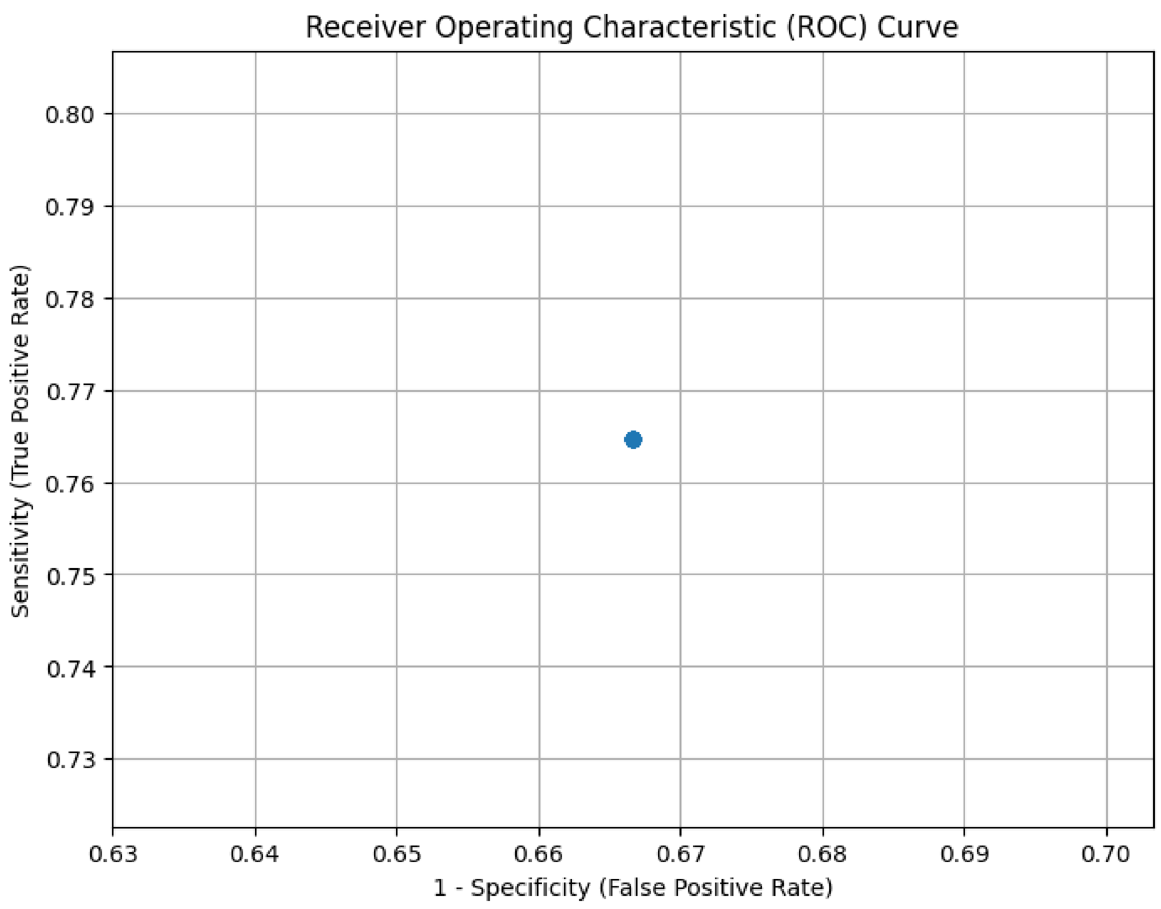
```
#3.What is the  specificity  of the detection method?[0.4 Points]
TN = 50
FP = 100

# Calculate specificity
specificity = TN / (TN + FP)

# Print specificity
print("Specificity (True Negative Rate):", specificity)
```

```
Specificity (True Negative Rate): 0.3333333333333333
```

In [ ]:

```
#4 Draw the point of ROC for the given parameters (as for the single setup of de
import matplotlib.pyplot as plt

# Corrected values for the confusion matrix
TP = 650
TN = 50
FP = 100
FN = 200

# Calculate sensitivity and specificity
sensitivity = TP / (TP + FN)
specificity = TN / (TN + FP)

# ROC Curve
thresholds = [0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]
sensitivity_values = []
specificity_values = []

for threshold in thresholds:
    TP = 650  # Reset TP for each threshold
    FP = 100
    TN = 50
    FN = 200

    sensitivity = TP / (TP + FN)
    specificity = TN / (TN + FP)

    sensitivity_values.append(sensitivity)
    specificity_values.append(1 - specificity)

# Plot ROC Curve
plt.figure(figsize=(8, 6))
plt.plot(specificity_values, sensitivity_values, marker='o')
plt.title("Receiver Operating Characteristic (ROC) Curve")
plt.xlabel("1 - Specificity (False Positive Rate)")
plt.ylabel("Sensitivity (True Positive Rate)")
plt.grid(True)
plt.show()

# Print sensitivity and specificity
```

```
print("Sensitivity (True Positive Rate):", sensitivity)
print("Specificity (True Negative Rate):", specificity)
```



```
Sensitivity (True Positive Rate): 0.7647058823529411
Specificity (True Negative Rate): 0.3333333333333333
```

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

No description has been provided for this