Faculté des Sciences appliquées

ELEN060-2 : Information and Coding Theory

Project 1 : Information measures

Staff :
WEHENKEL Louis, *Teacher*
CIOPPA Anthony, *Assistant*
LAMBRECHTS Gaspard, *Assistant*

Group :
LOUIS Arthur
SAULAS Adrien

March 15, 2023

Contents

1 Implementation

1.1 Question 1 : `entropy`

The mathematical function used to compute the entropy is the following :

$$H(X) = -\sum_i p(X=i)\log_2 p(X=i)$$

The formula is easily implemented in the notebook using the functions `np.log` and `np.sum` while being cautious about possible zeros in the `numpy` array.

To explain intuitively what is measured by the entropy, is to think as the entropy as a measure of uncertainty or surprise. In general, the entropy of a system is higher when there are more possible outcomes and less prior knowledge about the probabilities of those outcomes. When there is more prior knowledge, the entropy is lower. It can also be used to quantify the amount of information that is contained in a signal or message. The higher the entropy, the more information is contained in the signal or message.

1.2 Question 2 : `joint_entropy`

The mathematical formula for joint entropy is:

$$H(X,Y) = -\sum_i \sum_j P(X=i,Y=j)\log_2(P(X=i,Y=j))$$

The formula is easily implemented in the notebook using the functions `np.log` and `np.sum` while being cautious about possible zeros in the `numpy` array. These functions are applied after flattening the array using the function `ravel`.

By comparing the formulas of the `entropy` and `joint_entropy` we clearly see that they look a lot alike, the `joint_entropy` is simply computed by looping on all the tuples of variables and taking the entropy of joint probability of the variables. The computation of the joint probability is done in a single `np.sum` function as the elements of `Pxy` already represent the joint probabilities.

1.3 Question 3 : `conditional_entropy`

The formula used to compute the conditional entropy is :

$$H(X|Y) = -\sum_i \sum_j P(X=i,Y=j)\log_2(P(X=i|Y=j))$$

In the implementation we first need to compute the marginal distribution of Y by using the function `np.sum` on the axis 0 of the joint probability distribution `Pxy`. We then use the functions `np.log` and `np.sum` to compute the conditional entropy while being cautious about possible zeros in the `numpy` array.

Another way of computing the conditional entropy would be to compute the conditioal probability of X given Y for each value of Y and then use it to compute the entropy :

$$H(X|Y) = \sum_j P(Y=j)H(X|Y=j)$$

where

$$H(X|Y=y) = -\sum_i P(X=i|Y=y)\log_2(P(X=i|Y=y))$$

1.4 Question 4 : `mutual_information`

The formula used to compute the mutual information is :

$$I(X;Y) = \sum_i \sum_j P(X=i, Y=j) \log_2 \left(\frac{P(X=i, Y=j)}{P(X=i)P(Y=j)} \right)$$

The first step in our implementation is to compute the marginal probability distributions of X and Y using the `np.sum` on the axis 0 and 1 to create respectively `Py` and `Px`. The joint marginal probability distribution is then computed by taking the outer product of the two previous arrays using `np.outer`. We then compute the mutual information formula using the functions `np.log` and `np.sum` while being cautious of possible zeros in the `numpy` arrays.

The mutual information is a measure that represents the amount of knowledge a variable provides on another. If this measure is high, the two variables are strongly dependent to one another and that having knowledge provides a lot of information on the other one. On the contrary, if it is low, it shows that the two variables are independent to one another and that knowledge about a variable doesn't impact the knowledge we have on the other one.

1.5 Question 5 : `cond_joint_entropy` and `cond_mutual_information`

The mathematical formula used to compute the conditional joint entropy of X and Y given Z is :

$$H(X, Y|Z) = \sum_k P(Z=k)H(X, Y|Z=k)$$

This is implemented by computing the marginal probability distribution of Z and its entropy and then iterating over each value of Z and computing the joint entropy of X and Y for that value and then returning the difference between the total joint entropy and the entropy of Z.

The mathematical formula for the conditional mutual information of X and Y given Z is :

$$I(X;Y|Z) = H(X|Z) - H(X|Y, Z)$$

This is implemented in the notebook by forst computing the marginal probability distributions of X and Y and then iterating over each combination of X, Y and Z to compute the joint probability distribution. We then need to compute the conditional entropy of X given Z and the conditional entropy of X given Y and Z. We then just return the difference between the conditional entropy of X given Z and the conditional entropy of X given Y and Z.

2 Predicting the outcome of a football game

2.1 Question 6 : entropy of each variable

The values of the entropy's of all the different variables can be computed using the function we have just created and report them in the following table :

	entropy
H(outcome)	1.3348792529653017
H(previous_outcome)	1.4830023175176483
H(day)	2.806559657184559
H(time)	0.9325249591116449
H(month)	3.5826314181996297
H(wind_speed)	1.5847100094439006
H(weather)	1.7640836027093239
H(location)	0.9999389442845601
H(capacity)	1.5339149966727514
H(stadium_state)	0.6395467715690346
H(injury)	0.9998419902704185
H(match_type)	0.9999029333006982
H(opponent_strength)	1.5843542880706156

We can easily notice the correlation between the fact that the higher the cardinality of the variables the higher the entropy. This can be explained by the source coding theorem that states that the minimum number of bits required to represent a message from a source with a given entropy is equal to the entropy of the source multiplied by the length of the message. Higher cardinality resulting in higher entropy can be explained if you consider that the entropy is a measure of uncertainty, the entropy of a variable will then increase if the number of possible values it can take on increases, because the uncertainty will increase. This is reflected in the source coding theorem, because it tells us that a message from a source with higher entropy will need more bits to represent the message, as more values can be taken.

Moreover, the more the probability of the variables is evenly distributed between its possible values, the higher the entropy of this variable will be. Once again, this can be explained as the more a variable is evenly distributed between its possible values, the more the uncertainty will increase and vice-versa, if a variable has a value that appears a lot of the time, it will make the entropy decrease. An example could be the variables `stadium_state` and `injury`. The first one has the value `dry` 4189 times out of 5000 and the second has the value `yes` 2537 times out of 5000 and this results in it results in these variables to have respectively an entropy around 0.64 and 1, even though they have the same cardinality.

2.2 Question 7 : conditional entropy of `outcome`

	conditional_entropy
H(outcome|previous_outcome)	1.1814755551974467
H(outcome|day)	1.3334941322458804
H(outcome|time)	1.3338032007184812
H(outcome|month)	1.3303613323938615
H(outcome|wind_speed)	1.334727799689012
H(outcome|weather)	1.33383591648553
H(outcome|location)	1.3335129925165217
H(outcome|capacity)	1.3320215182015702
H(outcome|stadium_state)	1.3343243002115326
H(outcome|injury)	1.330242767276265
H(outcome|match_type)	1.3348306627351736
H(outcome|opponent_strength)	0.9386104485077148

We can compare the entropy's of `outcome` between the previous questions and when the entropy is conditioned by the variables `previous_outcome` and `wind_speed`. We can see that in the case of `previous_outcome` the entropy of `outcome` decreases. This can be understood as if you know the result of the previous match, the uncertainty of `outcome` decreases and therefore it is easier to predict it. On the contrary, we can notice that the entropy of outcome decreases by a much smaller factor when we know the `wind_speed`. We can interpret this as the knowing of the wind speed only influences the outcome of the match in a very small manner.

2.3 Question 8 : mutual information of the pair `month` and `capacity` and the pair `day` and `time`

By computing the asked mutual information we get the following results :

	`mutual_information`
I(`month`, `capacity`)	0.006068927667434791
I(`day`,`time`)	0.5046071260288234

This makes sense because the `month` of the game doesn't influence the `capacity` of a stadium but the `time` when a game is played during a day clearly influences the `day` of the game. Indeed a game played in the morning or the afternoon can only be played on a week-end day but a game played in the evening is much more likely to be played during the week days.

2.4 Question 9 : on which `outcome` to bet with data non-free

By computing the mutual information between the variable `outcome` and all the other variables, we get the following table :

	`mutual_information`
I(`outcome`,`previous_outcome`)	0.15340369776785479
I(`outcome`,`day`)	0.0013851207194213906
I(`outcome`,`time`)	0.0010760522468203503
I(`outcome`,`month`)	0.004517920571440142
I(`outcome`,`wind_speed`)	0.00015145327628988243
I(`outcome`,`weather`)	0.001043336479771722
I(`outcome`,`location`)	0.001366260448780049
I(`outcome`,`capacity`)	0.0028577347637310526
I(`outcome`,`stadium_state`)	0.0005549527537693491
I(`outcome`,`injury`)	0.0046364856890366846
I(`outcome`,`match_type`)	4.859023012811908e-05
I(`outcome`,`opponent_strength`)	0.39626880445758694

If we only have the funds to make our bets using one specific variable, using the `mutual_information`, we would choose the variable with the maximum value. This would result in betting using the variable `opponent_strength`, this variable having a `mutual_information` with value 0,396. If we had to make our choice using `conditional_entropy`, we would use the variable with the minimum value, looking in the table from Question 7, once again we would use the variable `opponent_strength` with a value of 0,9386.

2.5 Question 10 : on which `outcome` to bet with previous outcomes free

Now that the `previous_outcome` is known, we can compute the conditional mutual information between the variable `outcome` and all the other variables conditioned by `previous_outcome` to make our bet, we get the following table :

	conditional_mutual_information
I(outcome,capacity|previous_outcome)	0.004953992513573757
I(outcome,day|previous_outcome)	0.004737405783874049
I(outcome,injury|previous_outcome)	0.008997458391027058
I(outcome,location|previous_outcome)	0.002128508451425759
I(outcome,match_type|previous_outcome)	0.0005145649247342288
I(outcome,month|previous_outcome)	0.013689670050333058
I(outcome,opponent_strength|previous_outcome)	0.2445855018361326
I(outcome,stadium_state|previous_outcome)	0.0006226110403606544
I(outcome,time|previous_outcome)	0.0034734328084082833
I(outcome,weather|previous_outcome)	0.0030334910764067136
I(outcome,wind_speed|previous_outcome)	0.002120983416720623

Once again, if we wanted to bet using the information of the given previous_outcome and another variable, we would choose the variable opponent_strength as it has the maximum value for conditional_mutual_information with a value of 0,2446.

However, we can notice that the information provided by the opponent_strength variable is lower when conditioned by previous_outcome than in the previous question and it makes sense as the more information we get on a particular game, the more the entropy is going to decrease. Indeed, in our case, the fact that we know the previous_outcome reduces the uncertainty on the outcome of the game and makes the information given by the variable opponent_strength relatively less important.

2.6 Question 11 : Particularity of the home stadium using stadium_state and weather

By isolating the games that were played in the home team stadium we can conclude that the home team has some kind of protection (either they play inside or they have a roof above the stadium) because the weather doesn't influence the state of the stadium. This can be proven by computing the entropy of weather, stadium_state and their mutual information. We get the following values respectively : 1,74, 0 and 0. Which confirms that the stadium_state is always dry whatever the weather.

3 Playing with information theory-based strategy

3.1 Question 12 : Entropy in Mastermind

Since we have 6 possible colours identically distributed, we have for each peg one chance in 6 to fall on the right colour. So we just have to calculate :

$$H(X) = -\sum_i p(X=i)\log_2 p(X=i) = -6 * 1/6 * \log_2(1/6) = 2.584962500721156$$

to obtain the entropy of one slot, where i represents the different colours from 1 to 6

Moreover, since each colour can be repeated, each slot has the same probability of 1/6. We just have to calculate :

$$H(X) = -\sum_j \sum_i p(X=i)\log_2 p(X=i) = -4 * 6 * 1/6 * \log_2(1/6) = 10.339850002884624$$

to obtain the entropy of the entire code, where i represents the different colours from 1 to 6 and j represents the different slot from 1 to 4.

3.2 Question 13 : Entropy after first guess

Once we know that one of the four colors chosen is good and well placed we can then say that the entropy and therefore the uncertainty of this slot goes to zero. Moreover, if we know that we only have one correctly placed peg in our guess and no misplaced pegs, we know that the other colors in the guess can't be in the correct code, giving us more information on the possible codes to test but we still have to be careful to the cases were a peg is more than once in the guess.

The only possibilities for colors we have after the first guess depends on the color of the correctly placed peg. Indeed, if the correct peg is blue, we only have 3 more colors to test in the guesses, as blue cannot appear a second time since we have only a correct peg and no misplaced peg. If the correct peg is any of the other two colors, there are still 4 possible colors to test for the code.

We can use the probabilities of the correctly placed peg to weigh the probability of each slot :
If slot 1 is the good one :

$$P(\text{code}|\text{correct\_slot} = \text{slot\_1}) = \frac{1}{4} \times \frac{1}{4} \times \frac{1}{4}$$

If slot 2 is the good one :

$$P(\text{code}|\text{correct\_slot} = \text{slot\_2}) = \frac{1}{3} \times \frac{1}{3} \times \frac{1}{3}$$

If slot 3 is the good one :

$$P(\text{code}|\text{correct\_slot} = \text{slot\_3}) = \frac{1}{4} \times \frac{1}{4} \times \frac{1}{4}$$

If slot 4 is the good one :

$$P(\text{code}|\text{correct\_slot} = \text{slot\_4}) = \frac{1}{3} \times \frac{1}{3} \times \frac{1}{3}$$

Now that we have computed these probabilities, we can use the feedback to compute the conditional entropy of the game knowing this feedback :

$$H(\text{first\_feedback}) = 6.169925001442312$$

The information given by the first feedback is thus worth $10,33 - 6,17 = 4,16$ bits.

3.3 Question 14 : Entropy after another result

As for the previous question we will enumerate each possible case that is acceptable after receiving the second feedback telling that there's a correctly placed peg and a misplaced peg :

To simply implement this we can create a probability tree for each slot :

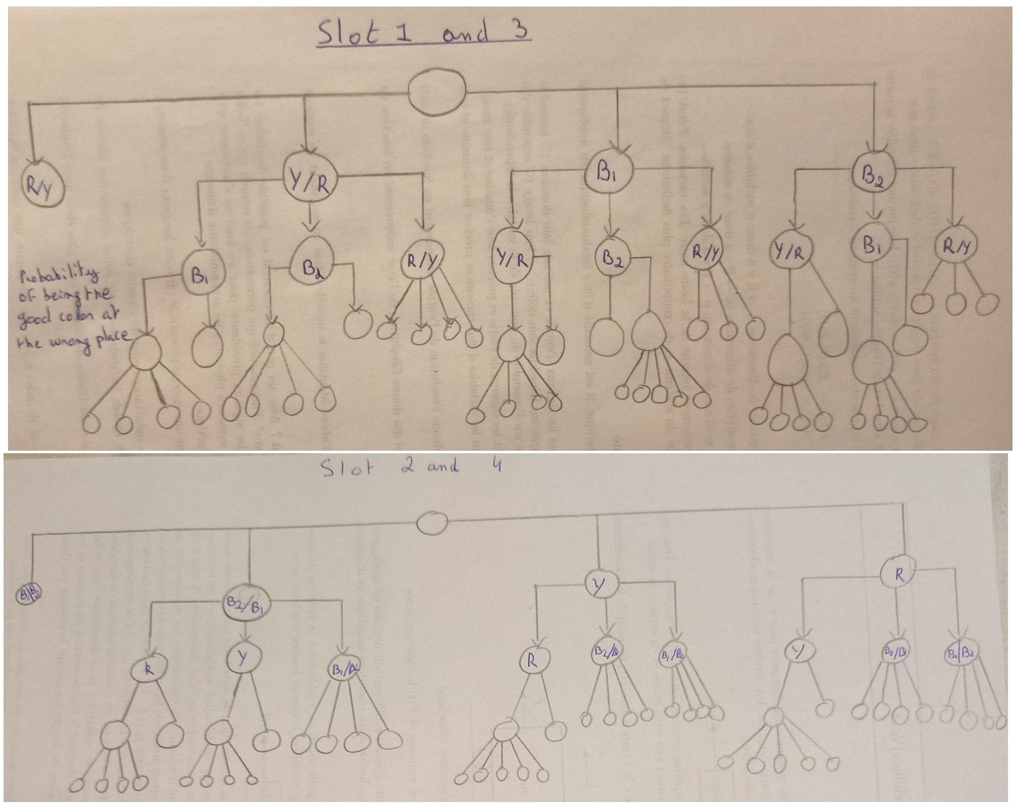

Figure 1: All possibility for the different slot

We can define each node from top to bottom:

1. first of all there are four possibilities depending on which peg is in the right color and in the right place
2. there are 3 possibilities according to which peg is in the good color but is not in the right place
3. there are two possibilities if the good colour that is not in the right place should be in the slot that is being watched or not
4. finally there are 3 to 5 possibilities left depending on which colors have been eliminated before

With all this we can distinguish two different cases:

$$P(\text{slot} = \text{slot1 or slot3}) = \frac{1}{4} \times \frac{1}{24} \times \frac{1}{36} \times \frac{1}{48} \times \frac{1}{96} \times \frac{1}{120}$$

$$P(\text{slot} = \text{slot2 or slot4}) = \frac{1}{4} \times \frac{1}{24} \times \frac{1}{48} \times \frac{1}{96} \times \frac{1}{120}$$

which allows us to find the following entropy:

$$H = 2 \times H(\text{slot} = \text{slot1 or slot3}) + 2 \times H(\text{slot} = \text{slot2 or slot4}) = 4.0213951381996385$$

which shows that knowing that a color is present but not in the right place decreases our uncertainty and therefore lowers our entropy, as expected. We can deduce the information provided by this feedback by simply computing $10,33 - 4,021 = 6,318$. Which proves us that the second feedback brings more information than the first one, as expected because we get more information on the system.

3.4 Question 15 : Maximum entropy formula

The formula of the maximum entropy can be found using the same principle as for the question 12 and by assuming that the secret codes are uniformly distributed to obtain the maximum entropy :

$$\begin{aligned} H_{\max} &= -\sum_{c \in \mathbf{C}^{\mathbf{S}}} P(C=c)\log_2(P(C=c)) \text{ c represents a combination from the possible } \mathbf{C}^{\mathbf{S}} \\ &= -\mathbf{C}^{\mathbf{S}} \times \frac{1}{\mathbf{C}^{\mathbf{S}}} \times \log_2(\frac{1}{\mathbf{C}^{\mathbf{S}}}) \\ &= \mathbf{S}\log_2(\mathbf{C}) \end{aligned}$$

The maximum entropy of the game grows with the number of possible slots and colors. Indeed, it makes the number of possible positions and thus the entropy grow. More specifically, the entropy grows linearly with the number of slots and logarithmically with the number of colors. It also makes sense because the number of possible positions grows much faster with the number of slots than with the number of colors.

3.5 Question 16 : Strategy based on information theory

To solve the Mastermind in the minimum number of guesses we could use a strategy based on the reduction of the entropy at each guess by trying to maximize the entropy reduction at each step.

To apply this, we first need to compute the entropy of the set of possible codes before making a guess. For each guess, we then compute the conditional entropy of the set of possible codes given the received feedback if that guess were chosen. We would then simply choose the guess that maximizes the expected reduction of entropy that is computed by taking the difference between the entropy before and after the guess and weighing everything by the probability of receiving a certain feedback after a specific guess.