Assignment 4
Neuroscience of Learning, Memory, Cognition
Dr. Karbalaei Aghajan

Mohammad Hossein Shafizadegan
99104781

June 18, 2023

Contents

1 PART I: Practical

categorizing the brain regions in the dataset

For each element in the "dat['brain_area']" array, we assign appropriate number to it using a switch case created by continuous if, else statements.

```
    for area in dat['brain_area']:
       if area in brain_groups[0] :
          barea.append(0)
       elif area in brain_groups[1] :
          barea.append(1)
       elif area in brain_groups[2] :
          barea.append(2)
       else :
          barea.append(3)
```

Plots by brain region

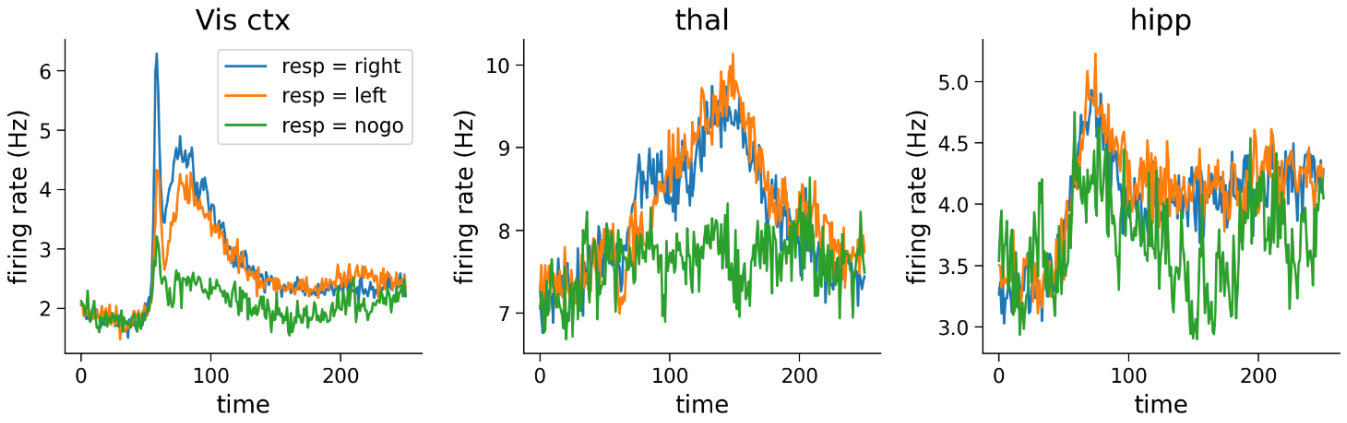
First we develop a function called "prepare_data()". The desired brain area (0:visual cortex, 1:thalamus, 2:hippocampus) will be provided as input argument. In this function we fist extract and separate data of this region of brain using the barea array we created before. Then we prepare data for each of the Left, right and nogo response categories as follows :

```
    response = dat['response'] # right - nogo - left (-1, 0, 1)
    barea_arr = np.asarray(barea, dtype=np.float64)
    area_data = dat['spks'][barea_arr == area_id, :, :]
    left_resp = area_data[:, response == 1,:]
    nogo_resp = area_data[:, response == 0,:]
    right_resp = area_data[:, response == -1,:]
```

Then we will create the "plot_data()" function. By parsing the input data to the function we will calculate the average through neurons and trials which are the first and second dimension of the input matrix. Finally we visualize the results.

```
    avg1 = np.average(data, axis=0)
    avg2 = np.average(avg1, axis=0)
```

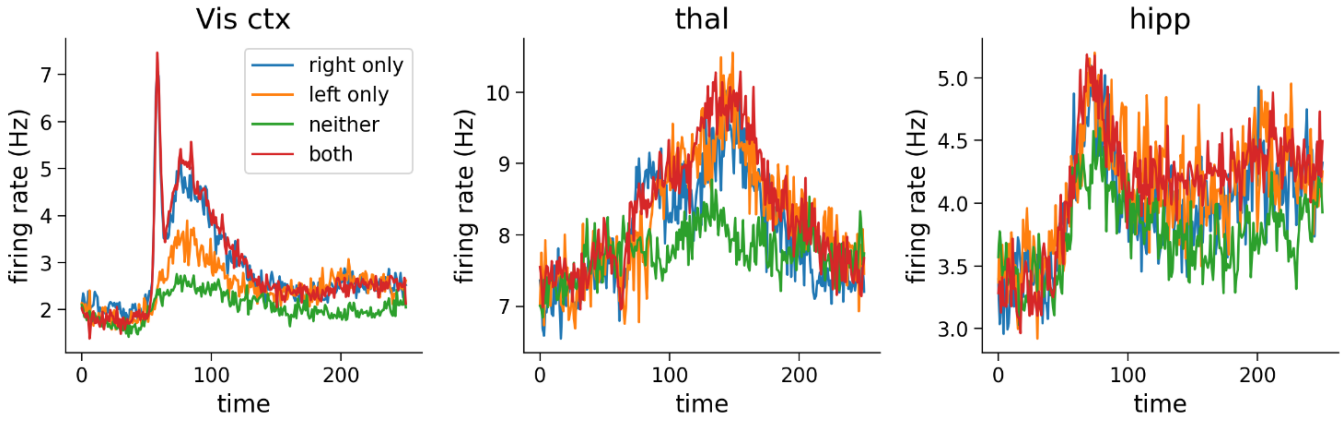
The final figures can be seen below :



Illustrating brain region activity with respect to visual conditions

The whole process in this section is quit the same as what we did in the previous section. Here we will regard and consider the right and left contrast when we want to separate the data for each categories. :

```
    vis_right = dat['contrast_right'] # 0 - low - high
    vis_left = dat['contrast_left'] # 0 - low - high
    barea_arr = np.asarray(barea, dtype=np.float64)
    area_data = dat['spks'][barea_arr == area_id, :, :]
    left_resp = area_data[:, (vis_left != 0)*(vis_right == 0),:]
    both_resp = area_data[:, (vis_left != 0)*(vis_right != 0),:]
    neither_resp = area_data[:, (vis_left == 0)*(vis_right == 0),:]
    right_resp = area_data[:, (vis_right != 0)*(vis_left == 0),:]
```



By observing these six figures it can be said that in average the neural activity as function of time is more dominant first in the visual cortex, then in the hippocampus and after all in the thalamus. This means that first the neurons of the visual cortex have been activated then the ones of the hippocampus and after that the neurons of the thalamus.

This change in activity in this order can show the visual data pathway and the dorsal pathway in the brain areas. The hippocampus, which is located in the medial temporal lobe, receives input from both streams and is involved in memory formation and spatial navigation.

The dorsal pathway is a visual system that processes spatial information and guides actions. The dorsal pathway starts from the primary visual cortex (V1) in the occipital lobe, where it receives input from the magnocellular pathway of the thalamus. The magnocellular pathway carries information about motion and depth from the retina.

The dorsal pathway is also known as the ”where” or ”how” pathway, because it is responsible for locating objects in space and guiding actions that depend on spatial perception. This explains the process of trial and the experiment taken on the rats that they have to choose the location of the stimulus on the right or left.

PCA (Principle Component Analysis)

We prepare data and run the PC analysis to find the principals components by following the step by step provided instructions as follows :

```
    NN = dat['spks'].shape[0] # number of neurons
    data = dat['spks'][:, :, 51:130]
    data = np.reshape(data, (NN, -1))
    data = data - np.mean(data, axis=1)[:, np.newaxis]
```

```
    data = data.T
    model = PCA(n_components=5) #run PCA and fit to data
    model.fit_transform(data)
    W = model.components_
    pc = W @ np.reshape(dat['spks'], (NN, -1))
    pc = np.reshape(pc, (5, -1, 250))
```

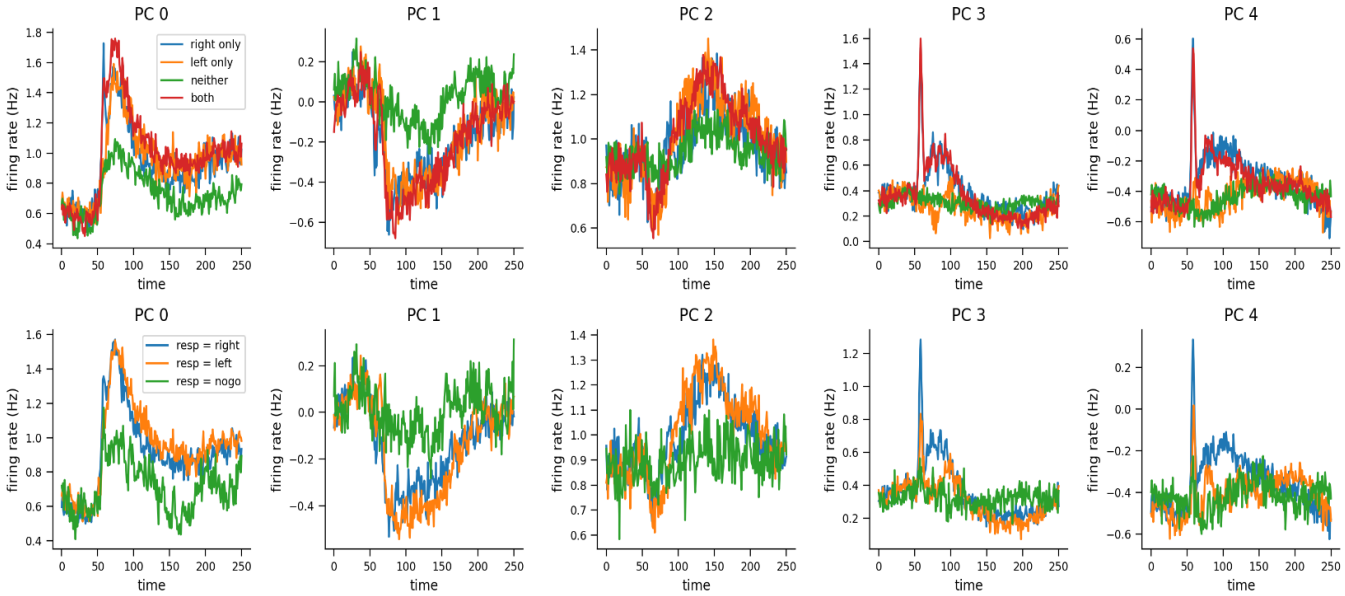
Now in order to plot the first 5 PCs with regards to visual conditions and response we have developed two functions. There is a function called "prepare_data_1". In this function we separate the input data which is one of the PCs, based on the type of response (left, right, nogo).

```
def prepare_data_1(data, title):
   response = dat['response'] # right - nogo - left (-1, 0, 1)
   left_resp = data[response == 1,:]
   nogo_resp = data[response == 0,:]
   right_resp = data[response == -1,:]
```

Also we have developed a function called "prepare_data_2". In this function we separate the input data with regards to the right and left contrasts.

```
def prepare_data_2(data, title):
   vis_right = dat['contrast_right'] # 0 - low - high
   vis_left = dat['contrast_left'] # 0 - low - high
   left_resp = data[(vis_left != 0)*(vis_right == 0),:]
   both_resp = data[(vis_left != 0)*(vis_right != 0),:]
   neither_resp = data[(vis_left == 0)*(vis_right == 0),:]
   right_resp = data[(vis_right != 0)*(vis_left == 0),:]
```

Finally we visualize all of the figures in a figure using "subplot"



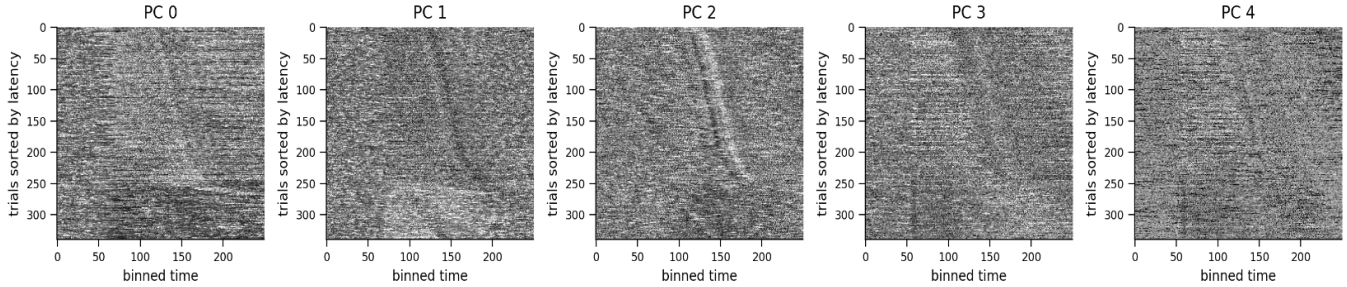
After that, we will sort trials with regards to latency. The results will be normalized using "zscore()" function imported from "scipy.stats" package. Finally we visualize the results using "imshow()" function with proper settings.

```
    isort = np.argsort(dat['response_time'].flatten())
    sorted_pc = pc[:, isort, :]
```

```
    for i in range(pc.shape[0]):
    sorted_pc[i] = zscore(sorted_pc[i])
```

Proper settings for "imshow" function :

```
    plt.imshow(data, aspect='auto', vmax=2, vmin=-2, cmap='gray')
```



Bonus (Extra) : Machine Learning techniques

We have used SVM technique to train a classifier which decodes and decides the rat response for two conditions left or right.
First we extract and choose the spike data related to the all left or right responses. Then we will define the "x" matrix as the input of the model. This matrix is total number of spikes for each of the 698 neuron. We respectively define the vector "y" which is the responses itself. Here we have 698 features which are the total number of spikes of neurons. The code of this section can be seen below :

```
    from sklearn.model_selection import train_test_split
    from sklearn import svm

    response = dat['response'] # right - nogo - left (-1, 0, 1)
    x = np.sum(dat['spks'][:, response != 0, :],axis=2)
    y = np.array(np.nonzero(response))
    y = np.reshape(y, (y.shape[1]))
    y = response[y]
```

Now we will use the "train_test_split()" function to randomly split our trails which are actually our samples. Then using the train data, we fit a SVM classifier with polynomial kernel and after that we are going to predict the result for the test data and calculate the accuracy of our model.

```
    x_train , x_test , y_train, y_test = train_test_split(x.T, y, test_size = 0.3)
    clf = svm.SVC(kernel='poly')
    clf.fit(x_train, y_train)
    y_predict = clf.predict(x_test)
    print("accurancy = ", np.sum(y_predict == y_test)/len(y_predict))
```

It can be seen that when our train data is 70 percent of our samples (trails) we will get the accuracy of about 94 % which is very good performance.

```
accurancy =  0.9397590361445783
```

Now we will improve our model by adding properties of regularization to avoid probable data over fitting. The code will be as follows :

```
pipeline = make_pipeline(StandardScaler(), svm.SVC(kernel='poly'))

hyperparameters = {'svc__C': [0.1, 1, 10], # Regularization parameter
                   'svc__degree': [2, 3, 4], # Polynomial degree
                   'svc__coef0': [0, 1, 2]} # Independent term in kernel function

grid_search = GridSearchCV(pipeline, hyperparameters, cv=5)
grid_search.fit(x_train, y_train)

best_model = grid_search.best_estimator_
best_score = grid_search.best_score_

test_score = best_model.score(x_test, y_test)
```

```
Best model: Pipeline(steps=[('standardscaler', StandardScaler()),
                            ('svc', SVC(C=1, coef0=1, degree=2, kernel='poly'))])
Best score: 0.9276653171390012
Test score: 0.9156626506024096
```

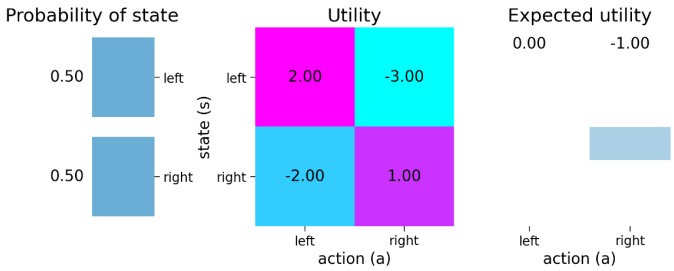
We observed that the performance of our model didn't improve so we can state the probably there was no over fitting for our model !

2 PART II: Theory

Interactive Demo 2: Exploring the decision

1

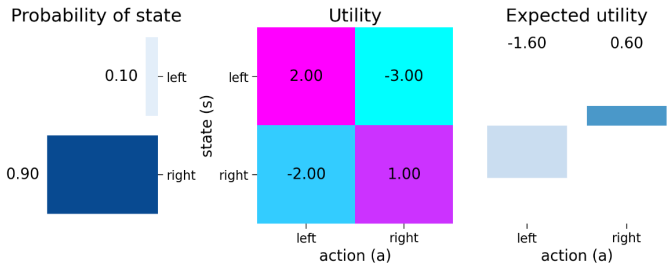
Since the submarine is on the right side of the dock, by assuming equal probabilities for the school being left and right, we will choose the left side for fishing.



It can be seen that the expected utility of the left side is higher than the right side as we expected.

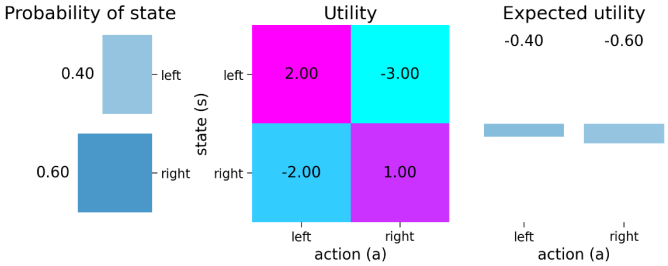
2

We saw that the utility of fishing on the left side is a bit higher than the right side but in this case the probability of the school being on the right side is much greater than the left side. This huge difference overcomes the slightly higher utility of the left side and now the expected utility of the right side is higher. Therefor, we choose fishing on the right side.



3

In this case although the probability of the right side is a bit higher but we won't choose the right side for fishing as the expected utility of the left side is still higher than the expected utility of the right side. We will choose the left side for fishing.



Think! 3: Guessing the location of the fish

1

$P(m = \text{catch fish}|s = \text{left}) = 0.7 \quad , \quad P(m = \text{no fish}|s = \text{left}) = 1 - 0.7 = 0.3$
$P(m = \text{catch fish}|s = \text{right}) = 0.2 \quad , \quad P(m = \text{no fish}|s = \text{right}) = 1 - 0.2 = 0.8$

2

$P(m = \text{catch fish}|s = \text{left}) > P(m = \text{catch fish}|s = \text{right}) \Rightarrow$ we would guess the school of fish is on the left side

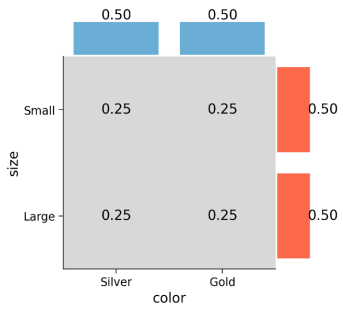
3

$P(m = \text{no fish}|s = \text{right}) > P(m = \text{no fish}|s = \text{left}) \Rightarrow$ we would guess the school of fish is on the right side

Think! 4.1: Covarying probability distributions

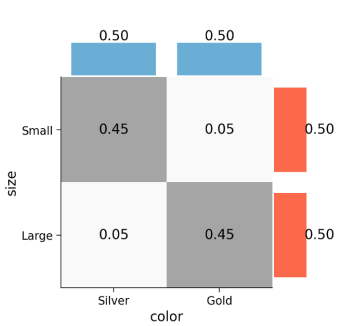
1

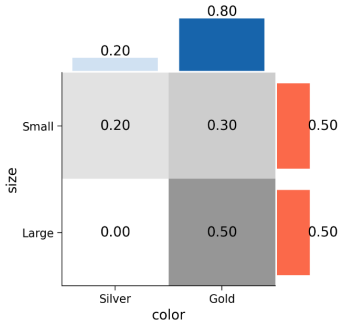
Having zero correlation means that the variables are completely independent and observing one of them won't give us any information about the other one. So the distribution of color tells nothing about the size in this case.



2

The correlation controls the distribution of probability across the joint probability table.
Both rows and columns need to sum to one, so for higher correlation, the probabilities are more restricted. The marginal probabilities show the relative weighting, the absolute probabilities for one quality will become more dependent on the other as the correlation goes to 1 or -1.

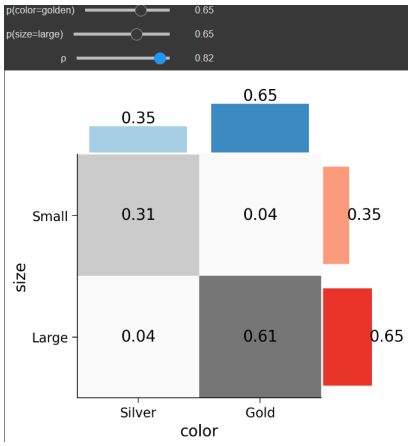




For the above tables, the correlation is equal to 0.8.

3

The correlation controls how much probability mass is located on the diagonals. By increasing the correlation, the probability of seeing the one of the two pairings has to goes towards zero.



4

As the absolute correlation increases, using the information we get from one quality, we will know more about the other quality.

Math Exercise 4.2.1: Computing marginal probabilities

1

$P(Y = \text{silver}) = P(X = \text{small}, Y = \text{silver}) + P(X = \text{large}, Y = \text{silver}) = 0.4 + 0.1 = 0.5$

2

$P(X = \text{small or large}, Y = \text{silver or gold}) =$
$P(X = \text{small}, Y = \text{silver}) + P(X = \text{large}, Y = \text{silver}) + P(X = \text{small}, Y = \text{gold}) + P(X = \text{large}, Y = \text{gold}) =$
$0.4 + 0.1 + 0.2 + 0.3 = 1$

3

$P(X = \text{small} or Y = \text{gold}) = P(X = \text{small}) + P(Y = \text{gold}) - P(X = \text{small}, Y = \text{gold})$
$P(X = \text{small}) = P(X = \text{small}, Y = \text{silver}) + P(X = \text{small}, Y = \text{gold}) = 0.6$
$P(Y = \text{gold}) = P(X = \text{small}, Y = \text{gold}) + P(X = \text{large}, Y = \text{gold}) = 0.5$
$\Rightarrow P(X = \text{small} or Y = \text{gold}) = 0.6 + 0.5 - 0.2 = 0.9$

Math Exercise 4.2.2: Computing marginal likelihood

1

$P(m = \text{fish}) = P(m = \text{fish}, s = \text{left}) + P(m = \text{fish}, s = \text{right})$
$= P(m = \text{fish}|s = \text{left})P(s = \text{left}) + P(m = \text{fish}|s = \text{right})P(s = \text{right}) = 0.1 \times 0.3 + 0.5 \times 0.7 = 0.38$

2

$P(m = \text{fish}) = P(m = \text{fish}, s = \text{left}) + P(m = \text{fish}, s = \text{right})$
$= P(m = \text{fish}|s = \text{left})P(s = \text{left}) + P(m = \text{fish}|s = \text{right})P(s = \text{right}) = 0.1 \times 0.6 + 0.5 \times 0.4 = 0.26$

Math Exercise 5: Calculating a posterior probability

1

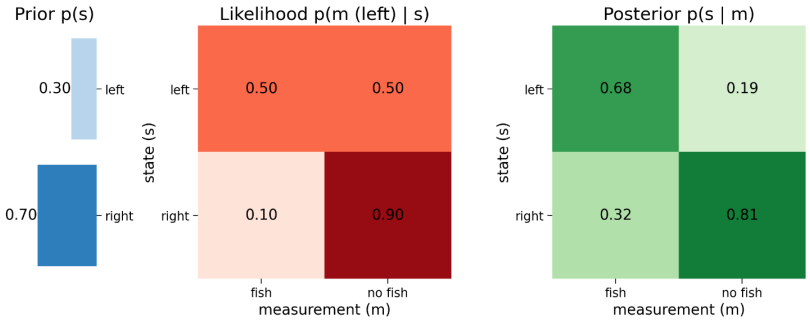
$P(m = \text{fish}) = P(m = \text{fish}|s = \text{left})P(s = \text{left}) + P(m = \text{fish}|s = \text{right})P(s = \text{right})$
$= 0.5 \times 0.3 + 0.1 \times 0.7 = 0.22$
$P(s = \text{left}|m = \text{fish}) = \dfrac{P(m = \text{fish}|s = \text{left})P(s = \text{left})}{P(m = \text{fish})} = \dfrac{0.5 \times 0.3}{0.22} = 0.68$

2

$P(m = \text{no fish}) = P(m = \text{no fish}|s = \text{left})P(s = \text{left}) + P(m = \text{no fish}|s = \text{right})P(s = \text{right})$
$= 0.5 \times 0.3 + 0.9 \times 0.7 = 0.78$
$P(s = \text{right}|m = \text{no fish}) = \dfrac{P(m = \text{no fish}|s = \text{right})P(s = \text{right})}{P(m = \text{no fish})} = \dfrac{0.9 \times 0.7}{0.78} = 0.81$

Extra (Bonus): Coding Exercise 5: Computing Posteriors

We simply calculate the unnormalized posterior by multiplying the likelihood and prior (likelihood × prior). Finally we normalize the result by dividing to p_m.

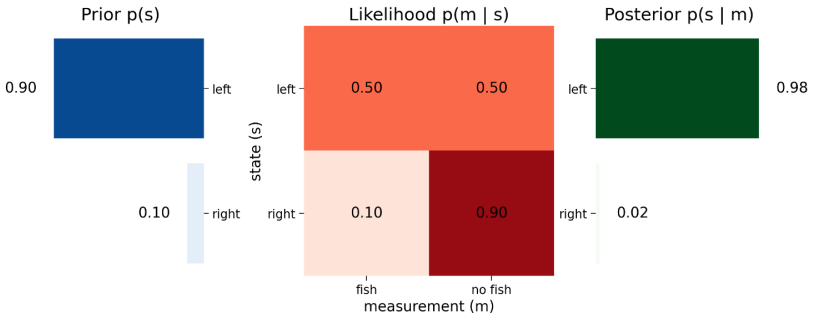


It can be clearly seen that the results match the calculations in the previous part.

Extra (Bonus): Interactive Demo 5: What affects the posterior?

1

When the prior is very informative, for example the prior that the fish are on the left side is 0.9, it strongly affects the posterior and the posterior probability of the state being left is high regardless of the measurement.



2

The differences between the likelihoods is a way of thinking about how much information we can gain. When the likelihoods are the same the information gained from catching a fish or not is less informative.

3

The likelihood exerts the most influence when it is informative, when catching a fish tells you a lot of information about which state is likely.

Section 6: Making Bayesian fishing decisions

1

There are actually many combinations that can produce the same expected utility for both actions, but the posterior probabilities will always have to balance out the differences in the utility function. So, what is important is that for a given utility function, there will be some 'point of indifference'

2

if the priors are quit the same and there is no significant difference, then the likelihood has more influence. Vice versa, if the likelihoods are quit the same , the prior is more important. Actually based on the Bayes rule, the relative information you gain from a measurement, and that you can use all of this information for your decision.

3

The model tells us how we should combine information and how we should act, based on some assumption about our goals. In this case, if we assume we are trying to maximize expected utility, we can state what an animal or subject should do.

4

Humans may not always try to maximize utility; humans and animals might not be able to calculate or represent probability distributions exactly; The utility function might be more complicated