### Review

Before we dive into code that utilizes Bayes' rule, let's review what we've learned in the previous lesson.

We've seen that Bayes' rule allows us to improve a **prior** probability by incorporating **new evidence** (from observed data or tests) and forming a new **posterior** probability. It does this through a series of mathematical steps.

To describe the steps, I'll be using the notation H*H* for hypothesis (ex. the likelihood that a car is in a certain location or that a person has cancer, etc.), and T*T* for observed test/sensor data (ex. a car sees the color green or a positive medical test result is returned). For example, P(T|\neg H)*P*(*T*∣¬*H*) is the probability of sensor reading occurring given that the hypothesis has not occurred.

**A note on notation**

As you work through these problems, you may see other notation, P(H| T)*P*(*H*∣*T*), or P(X| U)*P*(*X*∣*U*), or P(A| B)*P*(*A*∣*B*) (and so on) where one letter indicates a hypothesis and the other indicates observed data. Different notation; same concept, and as long as you are familiar with the concept, you should be well equipped to deal with changing notation!

### 1. Prior probabilities

The first step in Bayes' rule is to determine any prior probabilities. Ask yourself, based on previous data, how likely is a hypothesis, or specific event, H to happen?

* Determine *P*(*H*)
* Once you have P(H)*P*(*H*), you can derive  *P*(¬*H*)

### 2. Conditional/test probabilities

You should also know, through sensor or test data collection, how likely a certain test or sensor reading is to occur given that the hypothesis H has or has not occurred.

* Determine  *P*(*T*∣*H*) and  *P*(*T*∣¬*H*)
* Once you have P(T|H)*P*(*T*∣*H*), you can derive P(\neg T|H)*P*(¬*T*∣*H*)

Steps 1 and 2 give you all the information you need to perform Bayes' rule, and form a prediction about how likely a hypothesis is to be true given certain observed, related data.

### 3. Joint Probabilities

The next step is to calculate the four joint probabilities of the prior and the test probabilities. Two examples are given below.

* P(T, H) = P(T|H)\cdot P(H)*P*(*T*,*H*)=*P*(*T*∣*H*)⋅*P*(*H*)
* Likewise, P(T, \neg H) = P(T|\neg H)\cdot P(\neg H)*P*(*T*,¬*H*)=*P*(*T*∣¬*H*)⋅*P*(¬*H*)

### 4. Total probabilities

You'll then need to determine the total probability of a test result or sensor reading, so that you can use this value to normalize the posterior probability (which is the last step of Bayes' rule. The total probability of a test result is the sum of the joint probabilities in which that test result occurs. An example is given below.

* P(T) = P(T, H) + P(T, \neg H)*P*(*T*)=*P*(*T*,*H*)+*P*(*T*,¬*H*)

### 5. Posterior probability (last step)

The last step is to determine the probability of an event given a sensor reading or certain test data. And this is given by Bayes' rule. An example is shown below.

### Pattern Matching

Now that you know how to iterate through a 2D array, your next task will be to use what you've learned to help locate a robot in a 2D world based on some sensor measurements.

In this example, a robot is navigating a 4x5 orange and blue world, shown below. This robot also has a sensor that tells it the color of the square it's on top of and the color of the square directly to its right.

A screen shot of a building

Description automatically generated

2D orange and blue world grid

A screen shot of a computer

Description automatically generated

Robot sensing two color squares (one it's on top of, and one to its right)

In Python code, the world and the sensor measurement can be represented by the following code (where 'o' stands for orange and 'b' for blue):

**import** numpy **as** np

world = np.array ([ ['o', 'b', 'o', 'o', 'b'],

['o', 'o', 'b', 'o', 'o'],

['b', 'o', 'o', 'b', 'o'],

['b', 'o', 'o', 'o', 'o'] ])

measurement = ['b', 'o']

It's now easy for us to see that there are only a few places in the world that this robot can be, based on where the measurement lines up with color patterns on the world grid! In this exercise, it'll be up to you to find these locations programmatically.

**Hint**: You'll be adding values to a list. To add indices, please use the append function and the format list\_name.append([row,column]). Further instructions can be found in the TODOs below.

* [**pattern\_matching.py**](https://classroom.udacity.com/nanodegrees/nd113/parts/67914806-9163-4bef-85d4-5fa4a7409fbd/modules/ce9e580c-faf0-4f54-aeaf-6ff7ffd9c5d7/lessons/9986b101-885e-4ece-b522-b70a8b2cceed/concepts/4e8d09d7-6c6c-4ef3-baac-af3bdb5aac80)
* [**solution.py**](https://classroom.udacity.com/nanodegrees/nd113/parts/67914806-9163-4bef-85d4-5fa4a7409fbd/modules/ce9e580c-faf0-4f54-aeaf-6ff7ffd9c5d7/lessons/9986b101-885e-4ece-b522-b70a8b2cceed/concepts/4e8d09d7-6c6c-4ef3-baac-af3bdb5aac80)
* [**solution\_2.py**](https://classroom.udacity.com/nanodegrees/nd113/parts/67914806-9163-4bef-85d4-5fa4a7409fbd/modules/ce9e580c-faf0-4f54-aeaf-6ff7ffd9c5d7/lessons/9986b101-885e-4ece-b522-b70a8b2cceed/concepts/4e8d09d7-6c6c-4ef3-baac-af3bdb5aac80)