Assignment: Foundations of Bayes' theorem

Fill out the blanks as per the instructions below.

This assignment uses type hints, so make sure to stick to those.

Whenever you need to fill in a blank, we used Python's ellipsis (`...`).

Part 1 (A): Bayes' theorem with discrete random variables

Here, we assume a discrete prior $P(\theta)$, as well as a discrete probability distribution over a few i.i.d. observations.

The goal is to manually implement functionals for computing marginal and conditional likelihoods/probability densities.

You need to show that the posterior probability $P(\theta|Y)$ is a proper probability mass function. Choose a different number of parameters and observations.

In [13]:
```
# Let's define our parameters theta and their probabilities (our prior belief):
# A handful of thetas is enough.
theta: list[int] = [1,2,3]
theta_probs: list[float] = [0.2, 0.5, 0.3]   # sums to 1

# Here are our observations Y (don't change!):
Y_obs = [0.5, 1.2]

# Instead of assuming some (parameterized) distribution,
# we hardcode the conditional likelihoods of Y given some theta.
# Note that P(Y|\theta) is a likelihood, so it does not represent
# (necessarily) a valid probability density (i.e., values for each
# \theta do not necessarily have to sum to 1).
P_Y_given_theta: dict[int, dict[float, float]] = {
    1: {0.5: 0.10, 1.2: 0.40},
    2: {0.5: 0.30, 1.2: 0.20},
    3: {0.5: 0.05, 1.2: 0.60},
}
```

Define PMFs

For convenience, we define the PMFs for θ and Y explicitly:

In [14]:
```
# Don't change!
def P_theta(val: float) -> float:
    assert val in theta
```

```
    idx = theta.index(val)
    return theta_probs[idx]
```

Define Functions

for the likelihood $P(Y|\theta)$, the prior $P(\theta)$, and the evidence $P(Y)$.

Recall that the evidence:

$$P(Y) = \sum_i P(Y|\theta_i) \times P(\theta_i).$$

In [15]:
```
# Likelihood, P(Y|\theta), now explicitly from our discrete definition:
def likelihood(Y: list[float], t: float) -> float:
    lik = 1.0
    for y in Y:
        lik *= P_Y_given_theta[t][y]
    return lik

# The Evidence (in this assignment, it *is* computable):
def P_Y(Y: list[float]) -> float:
    temp = 0.0
    for t in theta:
        temp += P_theta(t) * likelihood(Y, t)
    return temp

# The posterior:
def P_theta_given_Y(t: float, Y: list[float]) -> float:
    temp = P_theta(t) * likelihood(Y, t) / P_Y(Y)
    return temp
```

In [16]:
```
# Don't change!
posterior_probs = [round(P_theta_given_Y(t=t, Y=Y_obs), ndigits=5) for t in thet
posterior_probs
```

Out[16]:
```
[0.17021, 0.6383, 0.19149]
```

In [17]:
```
# Don't change! The result here needs to be ~1.0!
print(sum(posterior_probs))
```

```
1.0
```

I add this markdown/LaTeX cell just to mathematically prove that the result that i got is true:

We can compute each likelihood:

$$P(Y|\theta = 1) = 0.1 * 0.4 = 0.04$$

$$P(Y|\theta = 2) = 0.3 * 0.2 = 0.06$$

$$P(Y|\theta = 3) = 0.05 * 0.6 = 0.03$$

We can then multiply the theese likelihoods with the probabilites of each outcome:

$$0.2 * 0.04 = 0.008$$

$$0.5 * 0.06 = 0.03$$

$$0.3 * 0.03 = 0.009$$

We can then sum theee together in order to get:

$$P(Y) = 0.008 + 0.03 + 0.009 = 0.047$$

Now we can finally compute the conditional probabilities, which means that we can compute the probability of a given theta value given that Y holds.

$$P(\theta = 1|Y) = \frac{0.08}{0.047} = 0.17021$$

$$P(\theta = 2|Y) = \frac{0.03}{0.047} = 0.63830$$

$$P(\theta = 3|Y) = \frac{0.009}{0.047} = 0.19149$$

This makes sence because of two reasons. Firstly we can see that the higher theta probabilites give higher conditional probabilites and this makes sence because the numerator in the conditional fraction will be higher but Y is constant. And the second reason is that when we sum all of theese together, they add up to 1. This is becuse when we add all outcomes together we get a summed probability of 1.

Part 1(B): Bayes' theorem with continuous random variables

Now, we change our model a bit. Instead of assuming a small discrete set of possible values for θ, we will assume that this parameter follows a standard normal distribution.

For our actual model, we will assume another normal distribution, where the standard deviation (scale) is fixed at $\frac{3}{2}$ and the mean is set to θ: $N \sim (\mu = \theta, \sigma = \frac{3}{2})$.

We will re-use the previous observations.

The evidence, defined continuously:

$$P(Y) = \int_{\theta} P(Y|t) \times P(t)\, d\theta.$$

In [18]:
```
#!pip install scipy
from scipy.stats.distributions import norm

# Our prior:
def P_theta_continuous(val: float) -> float:
    # Use norm.pdf() to compute this.
    norm_pdf = norm.pdf(val, loc=0, scale=1).item()
    return norm_pdf
```

In [19]:
```
import numpy as np
from scipy.integrate import quad
```

```
from typing import final


# Realistically, our bounds could be -10,10 (or similar), but
# scipy's quad allows to use infinity, so we'll use that, as
# it's also closer to how we would formulate this mathematically.
a, b = -np.inf, np.inf


# Our model prototype that takes a single scale parameter that
# will be held fixed for any subsequent likelihood computations.
@final
class Model:
    """Keep using this model class as-is, no need to change it."""
    def __init__(self, scale: float):
        self.scale = scale

    def likelihood(self, x: float, mean: float) -> float:
        return norm.pdf(x=x, loc=mean, scale=self.scale).item()


def likelihood_continuous(Y: list[float], t: float, model: Model) -> float:
    likelihood = 1.0
    for y in Y:
        likelihood *= model.likelihood(x=y, mean=t)
    return likelihood


def integrand_function(t, Y, model): # helper function for P_Y_continuous
    return P_theta_continuous(t) * likelihood_continuous(Y=Y, t=t, model=model)

# for this i had to create a function integrand_function becuase the quad functi
def P_Y_continuous(Y: list[float], model: Model) -> float:
    """Use quad() to integrate."""
    result = quad(func=integrand_function, a=a, b=b, args=(Y, model))[0]
    return result


def P_theta_given_Y_continuous(t: float, Y: list[float], evidence: float, model:
    temp = P_theta_continuous(t) * likelihood_continuous(Y=Y, t=t, model=model)
    return temp

# The goal of this function is to assert that our posterior is
# a valid probability density that sums/integrates to 1.
def P_theta_given_Y_continuous_integral(Y: list[float], model: Model) -> float:
    evidence = P_Y_continuous(Y=Y, model=model)
    func = lambda t: P_theta_given_Y_continuous(Y=Y, t=t, model=model, evidence=
    return quad(func=func, a=a, b=b)[0]
```

Now we show the amount of evidence, as well as that our posterior integrates to ≈ 1:
Also, we show the amount of (log-)evidence:

In [20]:
```
# Don't change. Prints the log-evidence, as well as its integral (should be ~1.0
from math import log
use_model = Model(scale=1.5)

log(P_Y_continuous(Y=Y_obs, model=use_model)),\
P_theta_given_Y_continuous_integral(Y=Y_obs, model=use_model)
```

Out[20]: (-3.1912461104301157, 0.9999999999999991)

Find and use a better model

Recall that our observations were fixed at $[0.5, 1.2]$ and we assumed our model would be a normal distribution with standard deviation $\sigma = \frac{3}{2}$.

In [21]:
```
Y_obs_arr = np.array(Y_obs)
Y_obs_arr.std().item(), Y_obs_arr.mean().item()
```

Out[21]: (0.35, 0.85)

However, we know that the standard deviation should likely be smaller to accommodate our data better. What we want to show here, is that a better model (here: same as previous but with a fixed standard deviation closer to 0.35) produces a larger **evidence**.

In [22]:
```
from scipy.optimize import minimize_scalar

# Use 'minimize_scalar' to find some optimal solution
optimal_scale = minimize_scalar(lambda s: -log(P_Y_continuous(Y_obs, Model(scale
 # <- fill in the blank here
```

In [23]:
```
# Don't change! Prints the log-evidence, as well as its integral (should be ~1.0
better_model = Model(scale=optimal_scale)

log(P_Y_continuous(Y=Y_obs, model=better_model)),\
P_theta_given_Y_continuous_integral(Y=Y_obs, model=better_model)
```

Out[23]: (-2.360448901825615, 1.0)

Short evaluation (write 1-2 sentences per):

1. How has the (log-)evidence changed using the optimal scale?
2. In Bayesian terms, what does this result mean?

Answers:

1. When we used a fixed scale of $\sigma = 1.5$ we found that $log(P(Y|\sigma)) = -3.1912461$. After optimizing the scale we found that $log(P(Y)) = -2.360489$. This means that the difference between the initial and the optimized log-evidence increased by about 0.83 units.

2. The marginal likelihood (what is called 'evidence' in the code) $P(Y)$ measures how well the model predicts the observed data when it integrates over all possible θ. the fact that $log(P(Y))$ goes up means that the optimized noise scale explains the data better under that scaling assumption. The observed Y value are more probable by a factor of $exp(0.83)$ than under the original $\sigma = 1.5$.