STAT 447 Assignment 5

Caden Hewlett

2024-02-09

Question 1: Sequential Updating

Consider a joint a joint probabilistic model given by

$$\theta \sim \rho, \text{ and } (x_i \mid \theta) \overset{iid}{\sim} \nu_\theta, \text{ where } i \in \{1, 2, \dots, n\}$$

where ρ is a prior distribution for the unknown parameter θ, and $\{x_i\}_{i=1}^n$ is a sequence of observations with conditional distribution ν_θ.

Part 1

Write down the posterior distribution of θ given $\{x_i\}_{i=1}^n$.

In a more verbose sense, let Θ be the random variable for the unknown parameter, and θ be the realization of this random variable under the proposed prior distribution. We can then write the following (*purely for nomenclature reasons*)

$$\rho = p_\Theta(\theta), \text{ where } p_\Theta(\theta) \text{ is the PMF/PDF given by } \rho$$
$$P(X = x \mid \theta) = p_{X|\Theta}(x, \theta), \text{ where } p_{X|\Theta}(x, \theta) \text{ is the PMF given by } \nu_\theta$$

With this in mind, we can write the posterior of θ given $\{x_i\}_{i=1}^n$, where we describe the event that $\{X_i\}_{i=1}^n = \{x_i\}_{i=1}^n$ using Bayes' Rule.

We will begin by using the most verbose notation possible, for complete clarity.

$$P(\Theta = \theta \mid \{X_i\}_{i=1}^n = \{x_i\}_{i=1}^n) = \frac{p_\Theta(\theta) P\big(\{X_i\}_{i=1}^n = \{x_i\}_{i=1}^n \mid \Theta = \theta\big)}{P\big(\{X_i\}_{i=1}^n = \{x_i\}_{i=1}^n\big)}$$

We start by considering the joint likelihood function.

$$P\big(\{X_i\}_{i=1}^n = \{x_i\}_{i=1}^n \mid \Theta = \theta\big) = P\big((X_n = x_n, X_{n-1} = x_{n-1}, \dots, X_1 = x_1) \mid \Theta = \theta\,\big)$$

Then, using intersections, we can write the likelihood as:

$$P\big(\{X_i\}_{i=1}^n = \{x_i\}_{i=1}^n \mid \Theta = \theta\big) = P\Big(\bigcap_{i=1}^n (X_i = x_i) \mid \Theta = \theta\Big)$$

Now, we use the following property of *iid* random variables

$$\text{I.I.D} \implies \forall (i \neq j) \in [1, n],\ (X_i \mid \Theta) \perp (X_j \mid \Theta) \implies \forall (i \neq j) \in [1, n], P(X_i \cap X_j \mid \Theta) = P(X_i \mid \Theta) P(X_j \mid \Theta)$$

Hence, we can write the likelihood as:

$$P\big(\{X_i\}_{i=1}^n = \{x_i\}_{i=1}^n \mid \Theta = \theta\big) = \prod_{i=1}^n \Big(P(X_i = x_i \mid \Theta = \theta)\Big) = \prod_{i=1}^n (\nu_\theta) = (\nu_\theta)^n$$

Which, as you can see, simplifies nicely to $(\nu_\theta)^n$.

Now, our expression simplifies a little bit to the following:

$$P(\Theta = \theta \mid \{X_i\}_{i=1}^n = \{x_i\}_{i=1}^n) = \frac{p_\Theta(\theta)\nu_\theta^n}{P\big(\{X_i\}_{i=1}^n = \{x_i\}_{i=1}^n\big)}$$

If we wished to, we could write the following proportionality directly to conclude:

$$p_{\Theta|X_{1:n}}(\theta, \{x_i\}_{i=1}^n) = \pi_n \propto \rho \cdot \nu_\theta^n$$

Or, letting normalizing constant $\mathcal{Z}_{1:n} = P\big(\{X_i\}_{i=1}^n = \{x_i\}_{i=1}^n\big)$, we can write:

$$\pi_n(\theta) = \frac{\rho \cdot \nu_\theta^n}{P\big(\{X_i\}_{i=1}^n = \{x_i\}_{i=1}^n\big)} = (\mathcal{Z}_{1:n}^{-1}) \cdot \rho \cdot \nu_\theta^n$$

Giving us a nice expression for the posterior, both in proportionality and equality according to a normalizing constant.

Part 2

Suppose now we get an additional data point x_{n+1} with the same conditional distribution ν_θ. Show that using the posterior from part 1 as the *prior* and data equal to just x_{n+1} gives the same posterior distribution as redoing part 1 with the $n+1$ data points.

We wish to show that:

$$\pi_{(n+1)}(\theta) = \frac{p_\Theta(\theta)P\big(\{X_i\}_{i=1}^{n+1} = \{x_i\}_{i=1}^{n+1} \mid \Theta = \theta\big)}{P\big(\{X_i\}_{i=1}^{n+1} = \{x_i\}_{i=1}^{n+1}\big)} = \frac{\pi_n(\theta)P(X_{n+1} = x_{n+1} \mid \Theta = \theta)}{P(X_{n+1} = x_{n+1})}$$

We'll evaluate each expression in turn.

With the first term, we can say directly that:

$$\text{LHS} = \frac{p_\Theta(\theta)P\big(\{X_i\}_{i=1}^{n+1} = \{x_i\}_{i=1}^{n+1} \mid \Theta = \theta\big)}{P\big(\{X_i\}_{i=1}^{n+1} = \{x_i\}_{i=1}^{n+1}\big)} = (\mathcal{Z}_{1:(n+1)}^{-1}) \cdot \rho \cdot \nu_\theta^{n+1} \propto \rho \cdot \nu_\theta^{n+1}$$

By the exact process used in **Part 1**.

More interestingly, we can expand the second term to as follows:

$$\pi_{(n+1)}(\theta) = \frac{\pi_n(\theta)P(X_{n+1} = x_{n+1} \mid \Theta = \theta)}{P(X_{n+1} = x_{n+1})} = (\mathcal{Z}_{n+1}^{-1})\pi_n(\theta)\nu_\theta$$

Then, substituting our expression for $\pi_n(\theta)$ from **Part 1**:

$$\text{RHS} = (\mathcal{Z}_{n+1}^{-1})\Big((\mathcal{Z}_{1:n}^{-1}) \cdot \rho \cdot \nu_\theta^n\Big)\nu_\theta = \big(\mathcal{Z}_{n+1}^{-1} \cdot \mathcal{Z}_{1:n}^{-1}\big)\rho\big(\nu^n \cdot \nu^1\big) = (\mathcal{Z}_{n+1} \cdot \mathcal{Z}_{1:n})^{-1}\rho \cdot \nu_\theta^{n+1} \propto \rho \cdot \nu_\theta^{n+1}$$

Now, consider the following Lemma which we will use without proof.

Lemma

Let $f(y)$ and $g(y)$ be probability distributions acting on the same space $y \in Y$. Let

$$\exists c \in \mathbb{R}^+ \text{ s.t. } \forall y \in Y, f(y) = c \cdot g(y) \implies f \text{ and } g \text{ describe equivalent distributions.}$$

In our specific case, we have:

$$(\mathcal{Z}_{1:(n+1)})^{-1}\text{LHS} = (\mathcal{Z}_{n+1} \cdot \mathcal{Z}_{1:n})^{-1}\text{RHS}$$

So, we can write:

$$\text{Let } c = \frac{(\mathcal{Z}_{n+1} \cdot \mathcal{Z}_{1:n})^{-1}}{(\mathcal{Z}_{1:(n+1)})^{-1}} = \frac{\mathcal{Z}_{1:(n+1)}}{\mathcal{Z}_{n+1} \cdot \mathcal{Z}_{1:n}}, \text{ then LHS} = c \cdot \text{RHS}$$

Which would imply that the two are equivalent under proportionality, and, assuming $c \in \mathbb{R}^+$, are equivalent under normalization.

Question 2: Bayesian Inference in the Limit of Increasing Data

We will use the tractability of the coin bag example to explore the behavior of the posterior distribution as the number of observations goes to infinity. Recall that its joint distribution is

$$p \sim \text{discrete}(\{0, 1/K, 2/K, \ldots, 1\}, \rho)$$
$$y_i | p \overset{\text{iid}}{\sim} \text{bern}(p), \quad i \in \{1, \ldots, n\}$$

Where $\rho = (\rho_0, \rho_1, \ldots, \rho_K)$ is the prior, where $\forall k \in [1, K]$ the proportion of coins of type k in the bag is ρ_k. For the sake of notation (and to avoid confusion with capital P), we will let p_{obs} be a realization of the random variable p.

Part 1

The following simulates the posterior for the above equation.

```
posterior_distribution = function(rho, n_heads, n_observations) {
  K = length(rho) - 1
  gamma = rho * dbinom(n_heads, n_observations, (0:K)/K)
  normalizing_constant = sum(gamma)
  gamma/normalizing_constant
}
```

Note, we can verify that passing the following gives our familiar 1/17 result from Assignment 1.

```
assign_1 = posterior_distribution(c(0, 0.5, 1),  3,  3)
# check
all.equal( 1/17, assign_1[2], tolerance = 2e-10)
```

```
## [1] TRUE
```

Part 2

Write a function `posterior_mean` that computes the posterior mean given the output of `posterior_distribution`.

We recall that the posterior mean is computed as follows:

$$\mathbb{E}(p \mid Y = y) = \sum_{\{p\,:\,\pi(p)>0\}} p\,\pi(p)$$

From our "original" coin flip example, we'd have $K = 2$ and

$$\mathbb{E}(p \mid Y = y) = \left(\frac{1}{K}\right)\pi(1) + \left(\frac{2}{K}\right)\pi(2) = \left(\frac{1}{2}\right) \cdot \frac{1}{17} + \left(\frac{2}{2}\right)\frac{16}{17} = \frac{33}{34}$$

Which we will use to test our function.

```
posterior_mean <- function(pi){
  K = length(pi)-1
  return(sum((0:K)/K*pi))
}
all.equal(posterior_mean(assign_1), 33/34)
```

```
## [1] TRUE
```

Further, we'd expect a posterior mean of 1 in the case where all the weight was put on the 1 ("always heads") coin.

```
posterior_mean(posterior_distribution(c(0,0,1), 5, 5))
```

```
## [1] 1
```

Part 3

Write another function called `simulate_posterior_mean_error`.

It will perform the following tasks:

a.) Simulate $p_{\text{true}} \sim \text{discrete}(\{0, 1/K, \ldots, K/K\}, \rho_{\text{true}})$, where ρ_{true} is supplied to the function.

This is done as follows (using the Assignment 1 coin flip example)

```
rho_true = c(0, 0.5, 1)/sum(c(0, 0.5, 1)); K = length(rho_true)-1
p_true <- DiscreteDistribution(supp = (0:K)/K, rho_true)
p_true_obs = simulate(p_true)
p_true_obs
```

```
## [1] 1
```

b.) Simulates $\{y_i\}_{i=1}^{n_{\text{obs}}} = \{y_1, y_2, \ldots, y_{n_{\text{obs}}}\} = y_{1:n_{\text{obs}}}$ conditional on the simulated p_{true}. We recall that $\forall i \in [1, n_{\text{obs}}], (y_i \mid p_{\text{true}}) \sim \text{bern}(p_{\text{true}})$, where it should be noted that \$n_{obs} = \$`n_observations` is supplied to the function.

```
n_observations = 3
n = n_observations
y_obs = rbern(n, p_true_obs)
y_obs
```

```
## [1] 1 1 1
```

c.) Calls `posterior_distribution` using and the simulated data. Note that ρ_{prior} is the final parameter to the function.

```
# in the coin flip, we assumed all were equally likely
rho_prior = c(0, 1/2, 1)
pi_obs = posterior_distribution(
  rho_prior, sum(y_obs), 3
)
pi_obs
```

```
## [1] 0.00000000 0.05882353 0.94117647
```

d.) Computes the posterior mean $\mathbb{E}(p \mid y_{\text{obs}})$, given the observed posterior $\pi_{\text{obs}}(p)$.

```
post_mean = posterior_mean(pi_obs)
post_mean
```

```
## [1] 0.9705882
```

e.) Compute the absolute error ε_{obs}

$$\varepsilon_{\text{obs}} = \left| p_{\text{true}} - \mathbb{E}(p \mid y_{1:n_{\text{obs}}}) \right|$$

.

```
epsilon_obs = abs(p_true_obs - post_mean)
epsilon_obs
```

```
## [1] 0.02941176
```

Then we combine this all into one process

```
simulate_posterior_mean_error = function(rho_true, rho_prior, n_observations){
  # step 1
  p_true = DiscreteDistribution(supp = (0:K)/K, rho_true)
  p_true_obs = simulate(p_true)
  # step 2
  n = n_observations
  y_obs = rbern(n, p_true_obs)
  # step 3
  pi_obs = posterior_distribution(rho_prior, sum(y_obs), n)
  # step 4
  epsilon_obs = abs(p_true_obs - post_mean)
  # returns error
  return(epsilon_obs)
}
simulate_posterior_mean_error(rho_true, rho_prior, n)
```

```
## [1] 0.4705882
```

Part 4

Now, we use the following setup, normalizing the given ρ values so it can work with the `DiscreteDistribution` function.

```
K = 100
rho_true = rho_prior = rep(1, K+1) / (K+1)
```

Applying it to each $n \in \{1, 2, 4, 8, 16, 32, 64\}$

```
n_obs_vector <- 2^(0:6)
```

Repeating each case $M = 1,000$ times.

```
set.seed(0904)
M = 1000
all_simulations =
  matrix(sapply(1:M, function(m){
       sapply(n_obs_vector,
        function(n){
         simulate_posterior_mean_error(rho_true, rho_prior, n)})}), nrow = M)
```

While the method of using `apply` logic is a nice vectorized operation to improve simulation time, our current matrix does not have the desired column order (it has a column for each n, and each row represents the simulation number.) Let's fix this.

```
experiment_results  <- matrix(nrow = 0, ncol = 3)
for(i in 1:length(n_obs_vector)) {
  n_sim_results <- cbind( n_obs_vector[i], 1:M, all_simulations[, i])
  experiment_results <- rbind(experiment_results, n_sim_results)
}

colnames(experiment_results) = c("n_observations", "replication", "errors")
experiment_results = data.frame(experiment_results)
rownames(experiment_results) = 1:(7*M)
```

Then we can display the `head` and `tail` of this data frame.

```
kable(head(experiment_results))
```

n_observations	replication	errors
1	1	0.0505882
1	2	0.7505882
1	3	0.8605882
1	4	0.4905882
1	5	0.6505882
1	6	0.4405882

```
kable(tail(experiment_results))
```

	n_observations	replication	errors
6995	64	995	0.6505882
6996	64	996	0.4205882
6997	64	997	0.6105882
6998	64	998	0.8905882
6999	64	999	0.1605882
7000	64	1000	0.4205882

The following checks that our transformation did not obscure or misplace any of the data.

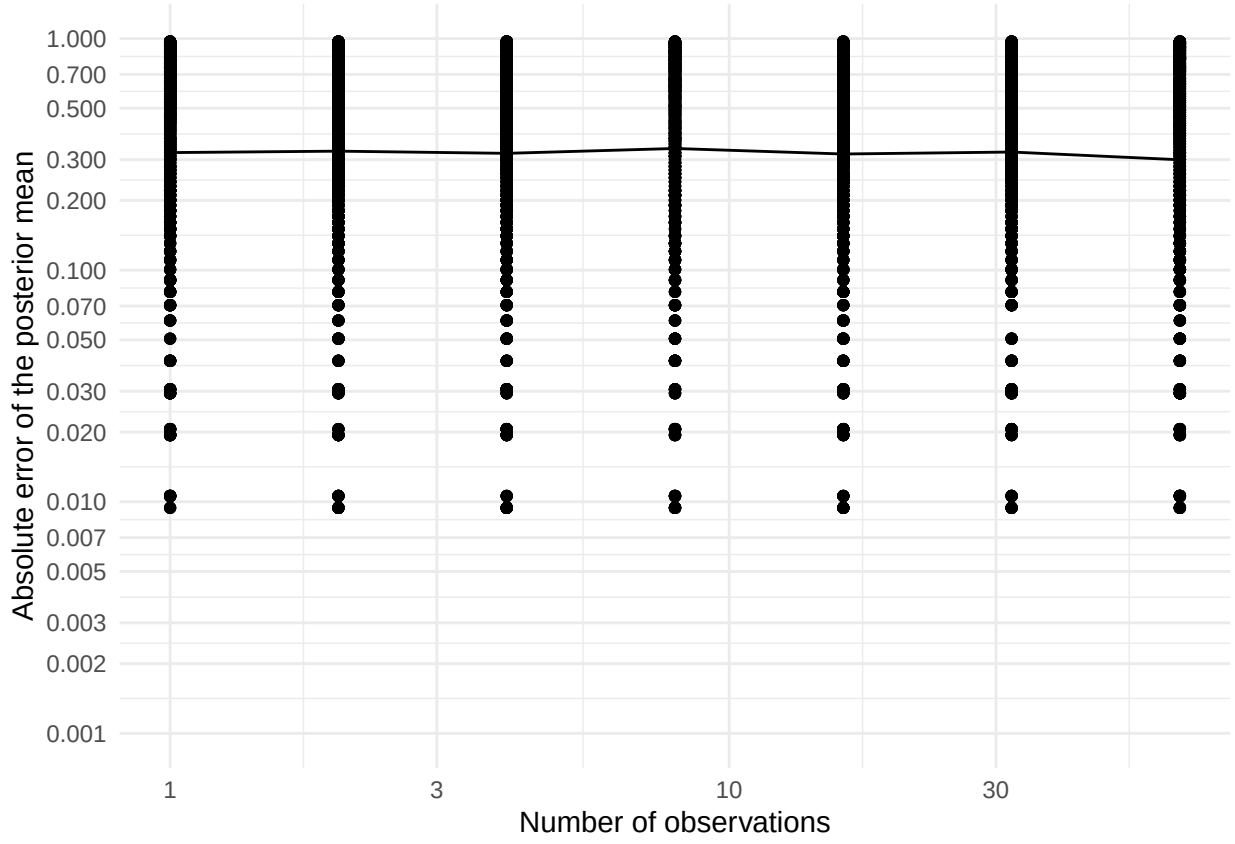
```r
sum((all_simulations[, 1]) -
      experiment_results$errors[experiment_results$n_observations == 1]) == 0
```

```
## [1] TRUE
```

Part 5

Visualize the above data using a log-log plot using the following code:

```r
ggplot(experiment_results, aes(x=n_observations, y=errors+1e-9)) + # avoid log(0)
  stat_summary(fun = mean, geom="line") + # Line averages over 1000 replicates
  scale_x_log10() +  # Show result in log-log scale
  scale_y_log10(n.breaks=16) +
  coord_cartesian(ylim = c(1e-3, 1)) +
  theme_minimal() +
  geom_point() +
  labs(x = "Number of observations",
       y = "Absolute error of the posterior mean")
```



Part 6

We'll estimate the slope with the following function, again letting ε be the error.

$$\hat{m} = \frac{y_7 - y_5}{x_7 - x_5} = \frac{\log_{10}(\bar{\varepsilon}_7) - \log_{10}(\bar{\varepsilon}_5)}{\log_{10}(n_7) - \log_{10}(n_5)}$$

Where, as shown above, x is $\log_{10}(n_i)$, is y is $\log_{10}(\bar{\varepsilon}_i)$, where $\bar{\varepsilon}$ is the mean absolute error of the posterior mean and the subscript indicates the column of points from left to right.

So, in this case, the 7th column

```
x7 = log10(n_obs_vector[7])
x5 = log10(n_obs_vector[5])
y7 = log10( mean(all_simulations[, 7]) )
y5 = log10( mean(all_simulations[, 5]) )

m = (y7 - y5) / (x7 - x5)
(m)
```

```
## [1] 0.005403119
```

What can you deduce about the scaling law of the error?

notes before I gtg break - slope is greater than zero (and nearly zero) - implies that increasing sample size has neglible effect on the error between ptrue and simulated values (E[p given data]) - makes sense, since rho post and rho true are in fact the same - perhaps produce a plot of means as well as error bands (also inspect non loglog) - oh shit tyto is messaging me