Bayesian Prediction

CS109L - Spring 2015

May 19, 2015

1 Integration

Write an `R` function called `MidpointRule` that approximates the integral of a function using the midpoint rule for Reimann sums. This function should take as arguments the function to integrate, the low and high ends of the range over which to integrate it, and the number of points to evaluate the function at for the midpoint rule. Test your function by integrating some simple functions for which you know the answer.

```
MidpointRule <- function(f, min, max, k, ...) {
  ## Numerically integrates a function using the midpoint rule.
  ## Args:
  ##   fun - the function to integrate
  ##   min - low bound for integration
  ##   max - high bound for integration
  ##   k - the number of points to evaulate the function at
  ##   ... - any additional arguments to the function being integrated
}
```

2 Bayesian Prediction

Brain, a renowned Stanford biologist, is studying learning in lab mice. His favorite lab subject, Pinky, loves cheese. Each day, Brain places Pinky in a maze and times how long it takes Pinky to find a hidden piece of cheese. The maze and Pinky's starting location is the same each day, but the location of the cheese determined at random. Brain has concluded that days when Pinky performs better than the day before and days where Pinky performs worse than the day before occur equally often.[1] However, Brain suspects that the direction of movement may depend to some extent on how Pinky performed the previous day.

Data on these changes is available in which a $+1$ indicates an upward move of Pinky's performance (i.e. a faster completion time), and a -1 indicates a downward move. We'll assume that the completion times are kept to such high precision that no change at all is very unlikely. For instance, the data might be:

$$+1, +1, -1, +1, +1, +1, -1, -1, -1, +1, +1$$

These values indicate that Pinky's performance went up in the first two days, then it went down, and so on and so forth. We might model the completion times $X_0, X_1, ...$ as random variables of a ***Markov process***,

[1] Here, we define performance to be the length of time it takes Pinky to find the cheese. Better performance correlates with less time spent wandering before the cheese is found.

where

$$P(X_0 = +1) = P(X_0 = -1) = 0.5$$
$$P(X_{i+1} = +1|X_i = +1) = P(X_{i+1} = -1|X_i = -1) = \theta$$
$$P(X_{i+1} = -1|X_i = +1) = P(X_{i+1} = +1|X_i = -1) = 1 - \theta$$

for $i = 0, 1, ...$ and with θ being an unknown model parameter between 0 and 1. A Markov process is one in which the random variable X_t at time t is conditionally independent of all other X_i (where $0 \leq i < t - 1$), given X_{t-1}. For time series data, we often make this simplifying assumption because it closely approximates the real relationship of X_t to the values that come before it, namely in our case that Pinky's performance yesterday has a much greater impact on his performance today than does his performance three weeks ago. In fact, his performance two days ago and earlier is completely negligible. Without the Markov assumption, we would have to model the conditional probability of observing X_t as jointly dependent on all signals before it, which we will say is mathematically (and computationally) intractable.

Brain thinks that θ might be close to 0.5, or that it might be close to 0 or 1, or with somewhat smaller probability, it might be anything else. To express this, he decides to use a prior distribution for θ whose density is proportional to the following:

$$\theta^{-\frac{1}{2}} \cdot (1 - \theta)^{-\frac{1}{2}} \cdot (0.2 + \exp(-100 \cdot (\theta - 0.5)^2)$$

Below is a plot of the unnormalized density, showing that it favors values near 0, 0.5, and 1. Note that the density goes to infinity as θ approaches 0 or 1.

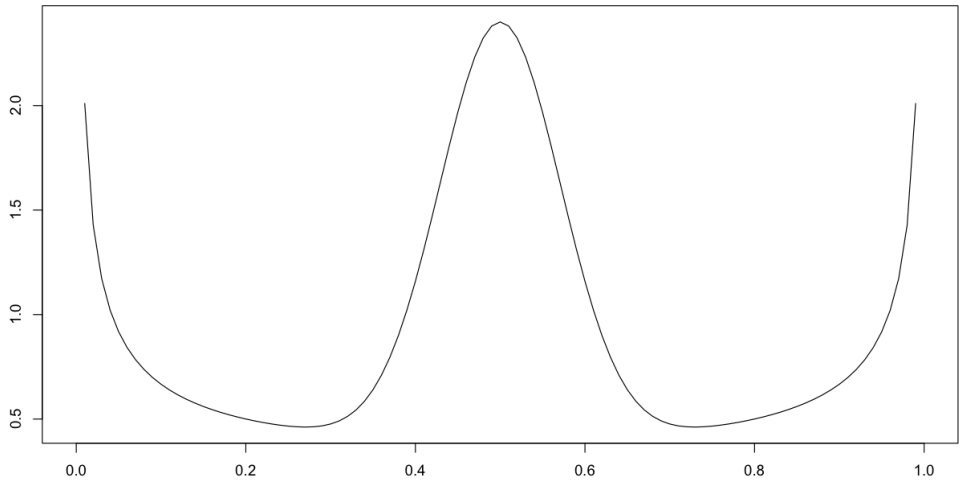

Figure 1: Bayesian Prior Density

Using the `MidpointRule` procedure you wrote in Part 1, answer the following questions:

1. What is the normalizing constant needed to convert the expression proportional to the prior density above to an actual prior density?
2. Given the eleven data points above, what is the maximum likelihood estimate (MLE) of θ?
3. How does the posterior distribution of θ given the eleven data points above compare to the prior distribution?

4. Given the eleven data points above, what is the Bayesian prediction for the next data point? What is the probability given to that prediction? The Bayesian prediction will be whichever value (+1 or −1) is given higher probability of being the next data point.

You should decide on an appropriate number of points to use for the midpoint rule by seeing at what point increasing the number of points makes little difference.

2.1 Instructions

Once you have completed `MidpointRule`, the boilerplate functions you should write next are `UPrior` and `PriorNorm`, which give the unnormalized prior density and the normalizing constant needed to make the prior density into a real probability distribution, respectively:

```
UPrior <- function(theta) {
  ## The unnormalized prior density function for the +1/-1 model. The valid
  ## range for theta is (0, 1) with the endpoints not included.
  ## Returns:
  ##   f(theta)
}

PriorNorm <- function(k) {
  ## Computes the normalizing constant for the prior density function.
  ## Args:
  ##   k - number of points to use for integration
}
```

The posterior distribution of θ is defined as

$$P(\theta|X) = \frac{P(X|\theta) \cdot P(\theta)}{P(X)}$$

where $P(\theta|X)$ is the posterior distribution, $P(\theta)$ is the prior distribution, and $P(X|\theta)$ is the "likelihood" (i.e. probability) of seeing the data X given θ. $P(X)$ is the normalizing constant needed to convert the posterior density into a real probability distribution. Your next step is to write the `Likelihood` and `UPost` functions, which compute the likelihood of data given θ and the *unnormalized* posterior distribution, respectively. Remember, the unnormalized posterior distribution omits the $P(X)$ term in the denominator above.

```
Likelihood <- function(theta, x) {
  ## The likelihood function. This is the probability of seeing x given theta.
  ## Args:
  ##   theta - distribution parameter
  ##   x - the data points (+1/-1 values)
}

UPost <- function(theta, x) {
  ## The unnormalized posterior density.
  ## Args:
  ##   theta - prior distribution parameter
  ##   x - the data points (+1/-1 values)
}
```

To find the maximum likelihood estimate of θ (that is, the value of θ which makes seeing our exact dataset most likely) plot the `Likelihood` function over a large number of θ values using the eleven data points above for `x` and find the value of θ where the `Likelihood` function is at a maximum. You can generate a smoothed plot of the `Likelihood` function using θ values from `seq(from = 0, to = 1, by = .01)`. From that vector of θ values and the `Likelihood` function, use `which.max` to find the index of the maximum likelihood estimate for θ. Using the same vector of θ values, you should also be able to plot a smoothed curve of the posterior distribution of θ which you should compare with the prior distribution.

Your final step is to write the `Predict` function, which predicts the value of the next data point. The function signature will look like this:

```
Predict <- function(x, k) {
  ## Predicts the next data point. Passed the previous data points and the
  ## number of points to use for integrations using the midpoint rule.
  ## Integrates the prediction for the next data point given theta with respect
  ## to the unnormalized posterior distribution of theta. Since the posterior
  ## is unnormalized, this results in unnormalized probabilities of the next
  ## point being +1 and -1, which are normalized with respect to the total
  ## unnormalized probability.
  ##
  ## Args:
  ##   x - the data points (+1/-1 values)
  ##   k - the number of points to use for integration
  ## Returns:
  ##   A list with fields $prediction and $probability. The $prediction field
  ##   is the value (+1/-1) predicted for the next data point (whichever has
  ##   higher probability of being the next data point). The $probability field
  ##   is the probability given to that prediction.
}
```

To complete the `Predict` function, you should compute the unnormalized probability of having the next data point be the same as the previous one and the unnormalized probability of having the next data point be different from the previous one, i.e $P(X_{n+1} = X_n \mid X)$ where X are all n original data. To do this, integrate (over all values of θ) the posterior distribution of θ times the probability of having the next data point be the same or different, respectively. The total density found in each case is proportional to the probability of having the next data point be the same or different. To get normalized probabilities for each of these events, divide unnormalized probabilities by the sum of the total densities. Whichever probability is larger will give you the prediction for the next data point.