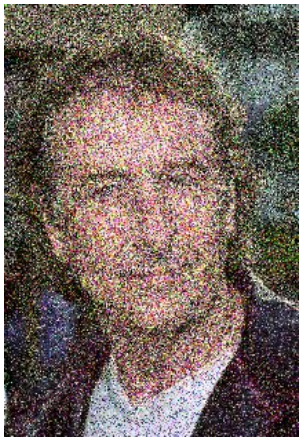
Computational Cognitive Science 2019-2020

Tutorial 3: Bayesian Estimation, Concept Learning

Question 1: Exercise in Bayesian Estimation

In an experiment on face recognition, subjects are presented with images of people they know, and asked to identify them. The images are presented for a very short period of time so that subjects may not have time to see the details of the entire face, but are likely to get a general impression of things like hair color and style, overall shape, skin color, etc. In this question we will consider how to formulate the face recognition problem as a probabilistic inference model.

1. What is the hypothesis space in this problem? Is it continuous or discrete? Finite or infinite?
2. What constitutes the observed data y and what kinds of values can it take on?
3. Write down an equation that expresses the inference problem that the subjects must solve to identify each face. Describe what each term in the equation represents.
4. What factors might influence the prior in this situation?
5. Suppose one group of subjects sees clear images, such as the one on the left below, and another group sees noisy images, such as the one on the right below. Which term(s) in your equation will be different for the noisy group compared to the clear group?

6. What does the model predict about subjects' performance with noisy images compared to clear images? Rather than working with the full scenario above, you can simplify by supposing the experiment has images of only two people, h_1 = Eric Idle and h_2 = John Cleese. How does image noisiness affect the model's probability of inferring that the image is of Eric Idle, rather than John Cleese? (Hint: since you are considering only two hypotheses, think about the *posterior odds*. How will this quantity differ between the case where the image is noisy and the case where it is clear?) Do you need to make any further assumptions in order to make predictions about subject behavior?

Question 2: A Bayesian Model of Concept Learning

In this question we will consider a simplified version of the Tenenbaum (2000) model of generalization where there are only four hypotheses under consideration, each of which has equal prior probability:

h_1 ={odd numbers}, h_2 ={even numbers}, h_3 ={multiples of 5}, h_4 ={multiples of 10}

Exercise: Given the set of examples X = {10, 40} from the target concept, compute the posterior probability of each hypothesis under the model.

The code block below has partly set up the problem, and indicates where you should add your own code.

Feel free to use more steps than shown below, e.g. defining intermediate variables or helper functions.

```
# identify the set of numbers, out of 100, defined by each hypothesis
h_set <- list(odd=seq(1, 99, by=2),
              even=seq(2,100, by=2),
              x5=seq(5,100, by=5),
              x10=seq(10,100, by=10))

# our example set:
X_set <- c(10, 40)
```

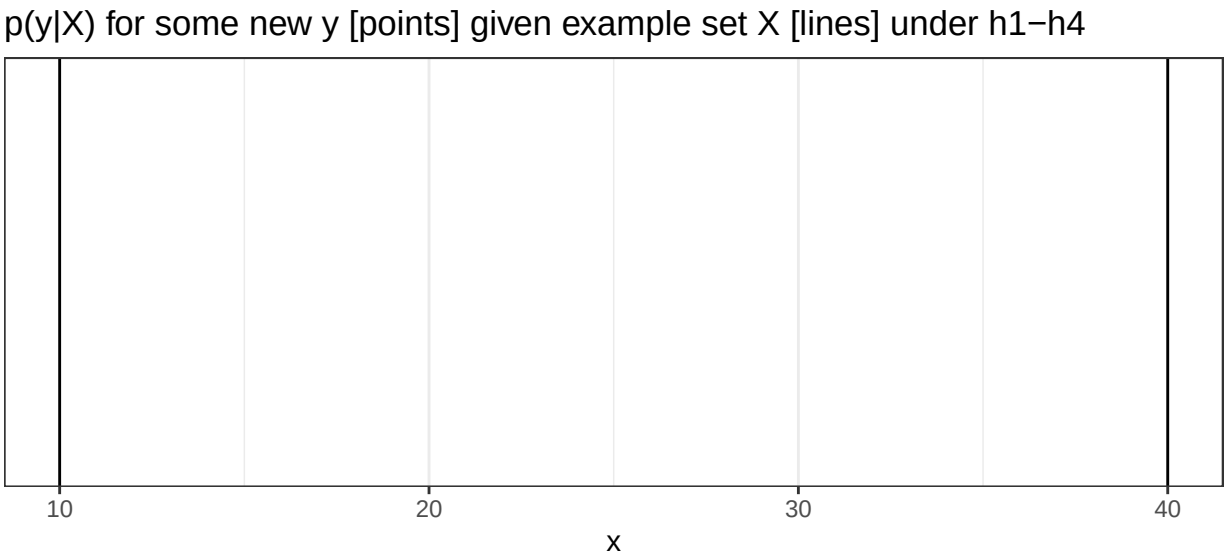
```
# prior:
#  make a vector with the prior probability for each hypothesis in h_set
p_h <- *YOUR CODE HERE*

# likelihood function:
#  write a function that takes inputs
#     X_set = the example set
#     h = one hypothesis (e.g. h_set[1])
#   and returns the likelihood p(X|h) as per Tenenbaum's Equation 1
likelihood <- function(X_set, h) { *YOUR CODE HERE* }

# calculate likelihood:
p_X_given_h <- sapply(h_set, likelihood, X_set=X_set)

# calculate posterior:
#  combine your prior p_h with the calculated likelihoods p_X_given_h
#  N.B. the result should be a probability
p_h_given_X <- *YOUR CODE HERE*

print(p_h_given_X)
```

Exercise: Determine the model's predicted probability that each of the following new data points is also part of the same concept: 2, 3, 5, and 20.

```
new_y <- c(2, 3, 5, 20)

# define function gen_probs,
#          to calculate the probablility that the example set X
#          will generalize to some new input y
#   inputs: y = one new data point, e.g. 2
#           all_h = h_set, the list of hypotheses defined above
#           all_p_h_x = p_h_given_X, the posterior probabilities you calculated above
#   return:
#         p(y in C | X), as per Tenenbaum's Equation 2
gen_probs <- function(y, all_h, all_p_h_x) { *YOUR CODE HERE* }

# calculate p(y|X) for each new input y

p_y_given_X <- sapply(X = new_y, # input to apply over
                      FUN = gen_probs, # function to apply
```

```
                      all_h = h_set, all_p_h_x = p_h_given_X) # additional named args

print(p_y_given_X)
```

```
# as in the last tutorial, we'll use ggplot to visualize.

# uncomment the section below once you've calculated p_y_given_X above

library(ggplot2)

ggplot() +
  geom_vline(data=data.frame(x=X_set),
             aes(xintercept=x)) +
  #geom_point(data=data.frame(new_x = new_y,
  #                           p_x = p_y_given_X),
  #           aes(x=new_x, y=p_x)) +
  ggtitle("p(y|X) for some new y [points] given example set X [lines] under h1-h4") +
  theme_bw()
```



Exercise: Note the plot above. Go back to the first cell and try to:

- add 50 to `X_set` and run everything again until the plot. What happens?
- add 30 to `X_set` and do it again.
- assign a new example set like so: `X_set <- c(15, 45)`, and run until the plot again. What's changed?
- add more examples to `new_y`. You can look at isolated numbers, or visualize the whole range.

Figure 1b from Tenenbaum (2000), reproduced below, shows model predictions for the sets listed (e.g. `X = 16`, `X = 16 8 2 24`, etc.).

Exercise: Run your code above using the example sets shown in Figure 1b (e.g. `X_set <- c(16)`). Do the plotted results look different from the figure? If so, why?

Exercise: Explain, with reference to the terms in the model, why the model's predictions tend to become more 'rule-like' as more examples are seen.

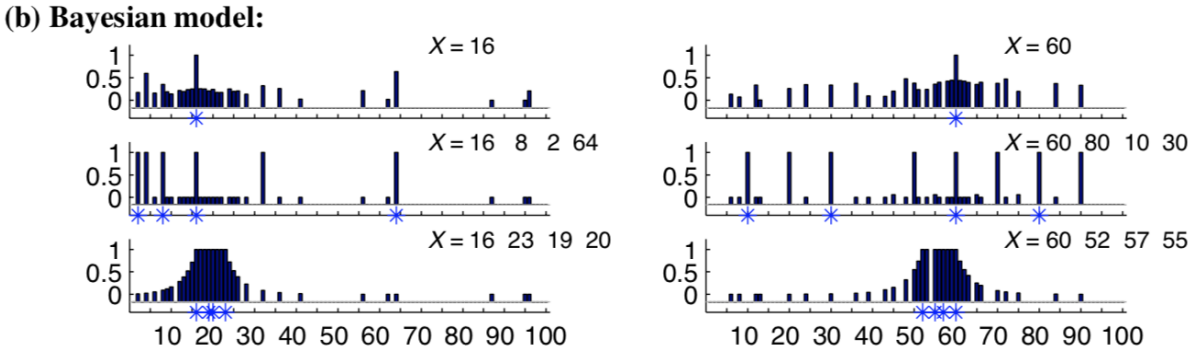


Figure 1: Tenenbaum predictions

References

Tenenbaum, Joshua B. (2000) "Rules and Similarity in Concept Learning," In *Advances in neural information processing systems*, 59–65.