EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Bayes II

Or: How to *actually use* Bayes theorem

- *The Language of Thought: computational cognitive science approaches to category learning*
- Who: Fausto Carcassi
- When: Sommer semester 2022

Where are we?

- Since the course started, we have been developing a series of concepts and techniques centering around the idea of a language of thought.
- We started with the philosophical idea of an LoT
- Then, we looked at how to model a fragment of the LoT for various conceptual domains, using PCFGs.
- Last time we started looking at Bayesian inference, with the aim of understanding how to learn sentences in the LoT from observations.
- Today we are going to keep looking at how to deal with an LoT probabilistically!

The problem: Bayesian evidence

- Last time we talked about Bayesian inference, but we didn't talk about how to do it in practice.
- Good old Bayes theorem:

$$P(H \mid D) = \frac{P(D \mid H)P(H)}{P(D)} = \frac{P(D \mid H)P(H)}{\sum_h P(D \mid h)P(h)}$$

- To calculate the denominator, we need to sum (or integrate) across all hypotheses. This is not possible except for the very simplest cases!
- E.g., consider: P(positive test | sick) = 0.9, P(positive test | not sick) = 0.1, P(sick) = 0.1. We can calculate P(sick | positive test).
- But in general, we need an alternative approach.

Note: We care about expectations

- The point here is that when it comes to analyzing the posterior distribution of a random variable X, we usually care about the expectation of a function of X, e.g. the mean or the variance.
- And therefore we can express our question about the posterior as a sum / an integral of a function of X.
- This is where a technique called *Monte Carlo Integration* is useful.
- Suppose we have a bunch of samples $x_1, \ldots, x_N$ from a distribution. Then:

$$\int f(x)P(x)dx \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i)$$

Monte Carlo integration

- Monte Carlo Integration means that to get any information we want from the posterior (e.g. mean, variance, histograms, etc.), all we need is *samples* from the posterior.
- Therefore, if we can get posterior samples, that's enough even if we can't calculate the full posterior probability.
- And it turns out that there's a (family of) really convenient algorithms to get samples from a probability even if all we have is a function that is just *proportional* to the distribution density function.
- The simplest algorithm of this type (which is used in Piantadosi's LOTlib3 library) is called *Metropolis-Hastings algorithm*.

Metropolis-Hastings algorithm

- Imagine you are on a ship on a lake
- You have a stick with which you can poke the bottom of the lake and determine its depth.
- Some parts of the lake are deeper than others and some more shallow.
- Problem: write down a list of points on the lake with a probability proportional to their depth.
- How would you go about doing this?
- Do you see why this is equivalent to the problem we have?

Metropolis-Hastings algorithm

- One solution:
- Start at any point $P_{current}$ at random
- Then for i=1; i < N; i++:
  - Move to a different point $P_{proposed}$ following a certain (symmetric) probability distribution centered at $P_{current}$
  - If depth($P_{proposed}$) > depth($P_{current}$):
    - Move to $P_{proposed}$, i.e. set $P_{current} = P_{proposed}$
  - Else:
    - Move to $P_{proposed}$ with probability depth($P_{proposed}$) / P($P_{current}$)
    - If they're almost the same, move with high probability, etc.
- Metropolis-Hastings is just this, but instead of depth we have probability!

Asymmetric proposal distribution

$$A(x', x_t) = \min\left(1, \frac{P(x')}{P(x_t)} \frac{g(x_t \mid x')}{g(x' \mid x_t)}\right)$$

Summary: Markov-chain Monte Carlo

- If some pretty weak conditions are satisfied, in the limit of infinite samples the distribution of samples converges to the true posterior distribution.
- We can think of MCMC as a way of getting samples from the posterior without knowing the normalization constant for the posterior, i.e. the *Bayesian evidence*.
- If we get enough samples, we can calculate an expectation of a function of the posterior with high accuracy, and therefore any 'summary' we are interested in.
- Now we have all the ingredients we need to apply Bayesian inference to cognitive models!

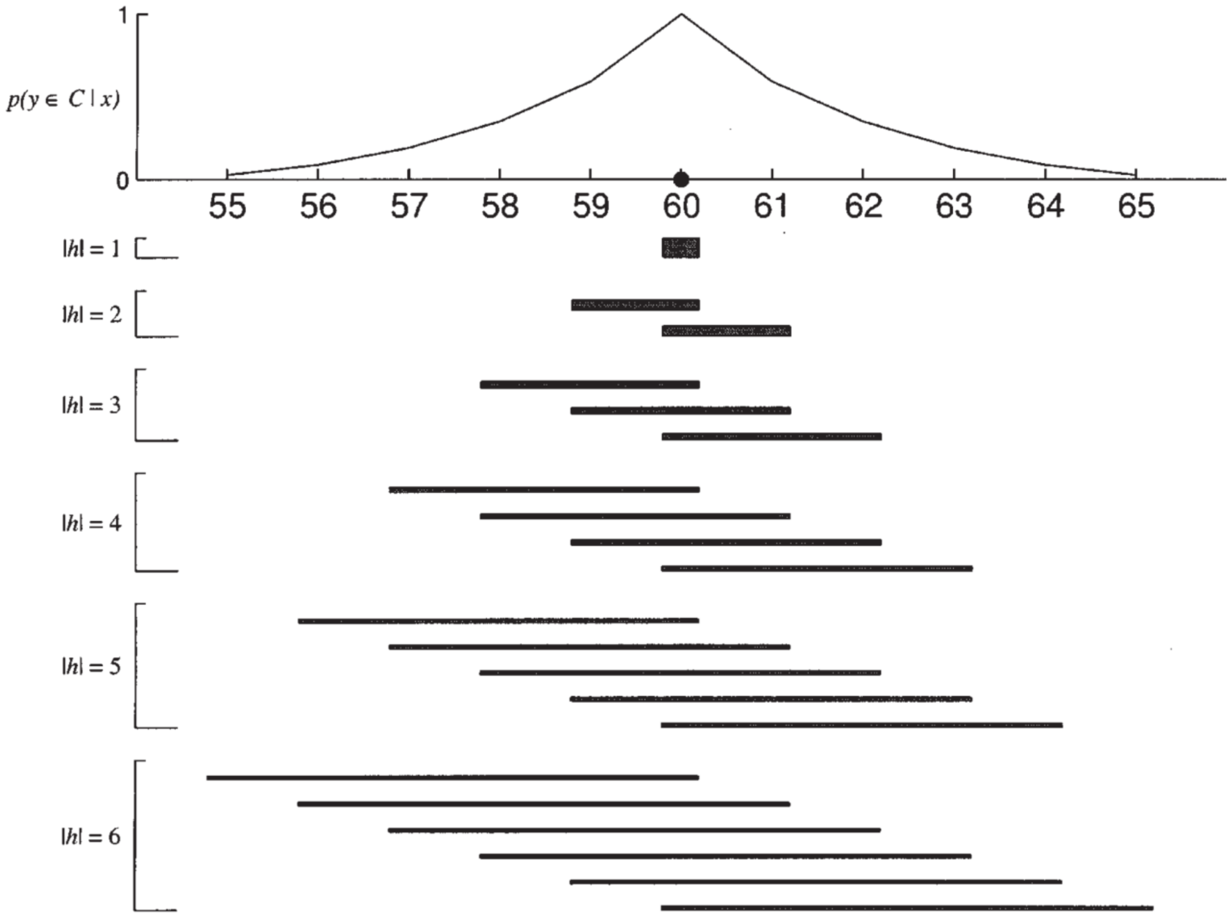
Case study: Simple category learning

- Suppose that we are trying to learn a category from examples.
- For simplicity, suppose that
  - The space is simply the integers from 1 to 50
  - The examples are numbers from the category
  - The category is *convex*, meaning we just need to set two borders
- We get examples from the category. There are two options:
  - *Weak sampling:* Both positive and negative evidence can be seen
  - *Strong sampling:* Only positive evidence can be seen

Simple category learning

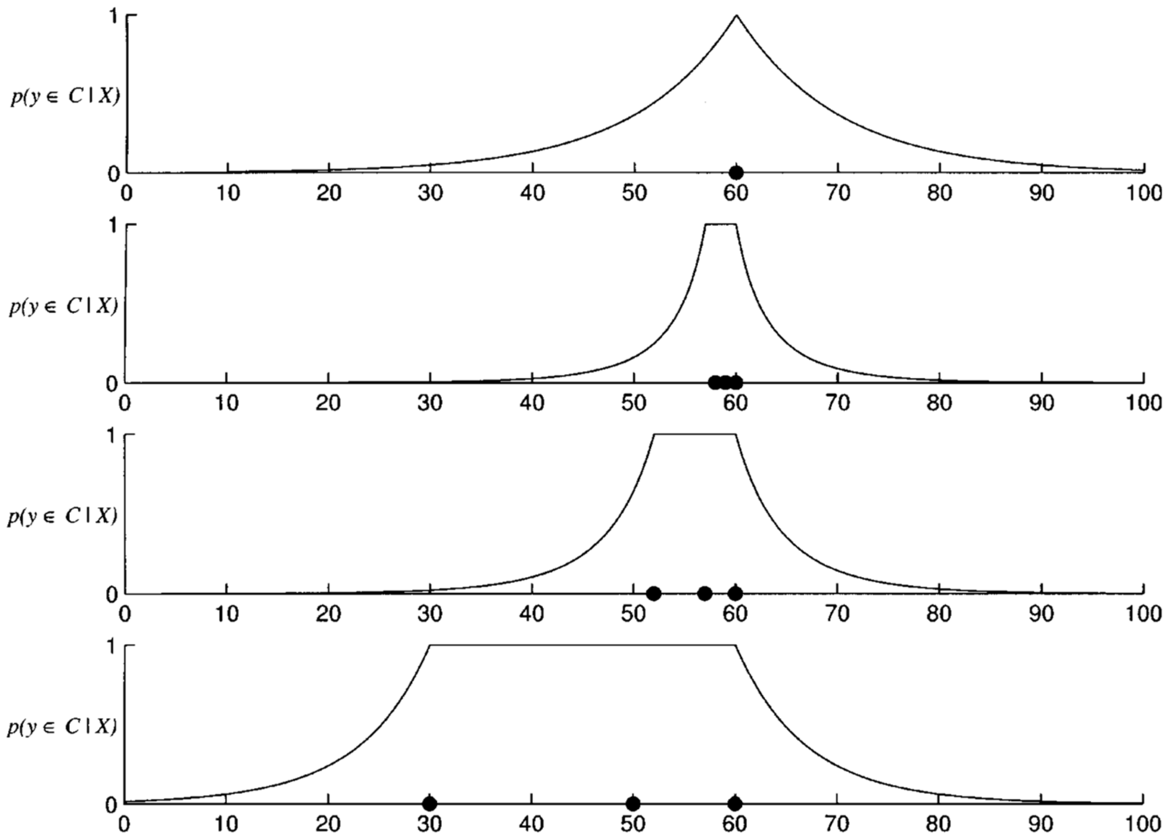
Let's go over this case of inference, assuming we got one observation!

- What's the space of hypotheses?
- What's the posterior, likelihood, and prior?
- What happens if we get more observations?



Simple category learning

- One important phenomenon here is the *size effect*
- More observations within a range makes the probability of the borders decrease faster.
- Can you see why formally?
- Can you see why intuitively?

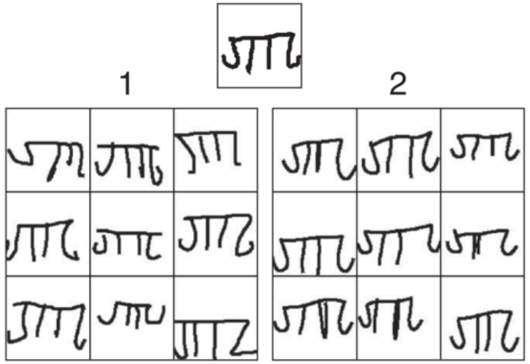


PCFGs and probabilities

- We have seen how to add probabilities to the production rules of a grammar, and we called those *probabilistic* context-free grammars.
- Basically, they give us the conditional probability of applying each rule given a certain nonterminal.
- This was the only point where probabilities enter the CFGs. However, we can also have probabilities at the level of the interpretation function.
- In this case, the interpretation of a sentence is not deterministic: evaluating a certain sentence multiple times can return different object.
- A sentence then returns a *distribution over objects* (in the relevant domain).

PCFGs and probabilities

- For instance, consider a fragment of the LoT that encodes handwritten characters. In a way, we can recognize the following as being 'the same character':



- The most natural way to make sense of this is to say that the same pLoT sentence can be realized in different ways, in virtue of it having a probabilistic component.
- In the context of learning an LoT expression from data, this gives us a likelihood P(data | LoT sentence).

Learning in a grammar

- Putting everything together, can we think of a way of sampling from the posterior distribution over sentences in an LoT given some observations?
- Prior, likelihood, proposal distribution